Diplomarbeit von Lennard Bernet

Digitale Plattform zur Trainingsverwaltung

Schule	TEKO Zürich
Diplomand	Lennard Bernet
Ausbildung	Dipl. Informatiker HF, *Applikationsentwicklung*
Start Ausbildung	Oktober 2023
Abgabe	28.10.2026
Klassifizierung	Öffentlich
Diplomcoach	Dominic Merz
Diplomexperte	Milad Fakiry

1 Management Summary 9
2 Initialisierung 10
2.1 Projektbeschrieb 10
2.1.1 Ausgangslage 10
2.1.2 Projektumfang und Zieldefinition 10
2.1.3 Projektvision 11
2.2 Projektmanagement 12
2.2.1 Makro-Management nach Hermes 12
2.2.2 Mikro-Management mit Kanban 12
2.3 Projektrisikomanagement 12
2.3.1 Risikoerkennung 13
2.3.2 Risikoanalyse 13
2.3.3 Risikomatrix 14
2.3.4 Massnahmen 15
2.4 Stakeholder Analyse 16
2.4.1 Stakeholder 17
2.4.1.1 Entwickler 17
2.4.1.2 Diplomcoach & Diplomexperte 17
2.4.1.3 Schule 17
2.4.1.4 Hosting und Infrastruktur Provider 18
2.4.2 Bewertung der Stakeholder 18
2.5 Wirtschaftlichkeitsanalyse 19
2.5.1 Buy-Case 19
2.5.2 Make-Case 19
2.5.3 Opportunitätskosten 20
2.5.4 Betriebskosten 20
2.5.5 Fazit 20
3 Planung 21
3.1 Makro Planung 21
3.1.1 GANTT Diagramm 22
3.1.2 Meilensteine 22
3.2 Mikro Planung 22
3.2.1 Arbeitspakete 22
3.2.1.1 Initialisierung 23
3.2.1.2 Planung 24
3.2.1.3 Vorbereitung 24

3.2.1.4 Anforderungsanalyse 25
3.2.1.5 Konzeption 25
3.2.1.6 Entwicklung 26
3.2.1.7 Deployment 27
3.2.1.8 Abschluss 28
3.2.1.9 Übersicht der Arbeitspakete 28
4 Vorbereitung/Analyse 29
4.1 Vorkenntnisse 29
4.2 Definition der Systemarchitektur 30
4.2.1 Architekturtreiber 31
4.2.2 Systemkontext 31
4.2.3 Entscheidungen 32
4.2.3.1 E1 - Persistenzansatz 33
4.2.3.2 E2 – Backend-Stil 34
4.2.3.3 E3 – Frontend-Ansatz 35
4.2.3.4 E4 - Deployment-Topologie 37
4.2.4 Bausteinsicht 38
4.2.5 Querschnittsthemen als Leitplanken 39
4.2.5.1 Fehlerbehandlung 39
4.2.5.2 Logging 39
4.2.5.3 API-Stil 39
4.2.5.4 Konfiguration 39
4.2.5.5 Validierung 39
4.2.6 Zusammenfassung der Leitplanken 40
4.3 Technologie-Evaluation 41
4.3.1 Entscheidungen 41
4.3.1.1 E5 – Backend-Framework 41
4.3.1.2 E6 – Frontend Framework 42
4.3.1.3 E7 – Hosting Provider 43
4.3.2 Ergänzende Technologien 44
4.3.3 Übersicht Technologie-Stack 46
4.4 Cyber Security Konzept 46
4.4.1 Schutzmassnahmen nach OWASP TOP 10:2025 47
4.4.2 Schutzobjekte und Schutzbedarf 47
4.4.3 Sicherheitszonen 48
4.4.4 Authentifizierungsverfahren 48

4.4.5 Autorisierung 48
4.4.6 Passwörter und Login 49
4.5 Datenschutz und rechtliche Aspekte 49
4.5.1 Bearbeitende Personendaten 49
4.5.2 Umsetzung der Grundsätze 50
4.5.3 Löschkonzept 50
4.5.4 Rechte der Benutzer und Datenschutzerklärung 50
4.6 Teststrategie 50
4.7 Versionsverwaltung 51
4.8 CI/CD-Pipeline 52
4.8.1 Werkzeuge 52
4.8.2 Umgebungen 52
4.8.3 Ablauf der Pipeline 52
4.9 UI/UX-Strategie 52
5 Anforderungsanalyse 54
5.1 Geschäftsanwendungsfälle 54
5.1.1 GAF1 Benutzerkonto führen 55
5.1.2 GAF2 Benutzerkonto auflösen 56
5.1.3 GAF3 Trainingsprogramm zusammenstellen 56
5.1.4 GAF4 Training planen 57
5.1.5 GAF5 Training durchführen und protokollieren 57
5.1.6 GAF6 Trainingsfortschritt nachvollziehen 57
5.1.7 Ausgeschlossene Geschäftsanwendungsfälle 58
5.2 Systemanwendungsfälle 59
5.2.1 SAF1 Registrieren 60
5.2.2 SAF2 Anmelden 61
5.2.3 SAF3 Profil verwalten 62
5.2.4 SAF4 Konto löschen 63
5.2.5 SAF5 Übung verwalten 63
5.2.6 SAF6 Trainingsroutine verwalten 64
5.2.7 SAF7 Training einplanen 65
5.2.8 SAF8 Training durchführen und protokollieren 66
5.2.9 SAF9 Trainingshistorie einsehen 67
5.3 Anforderungen 68
5.3.1 FT1 Benutzerverwaltung & Authentifizierung 69
5.3.2 FT2 Übungskatalog 72

5.3.3 FT3 Trainingsroutinen ... 73
5.3.4 FT4 Trainingsplanung ... 74
5.3.5 FT5 Live-Protokollierung ... 75
5.3.6 FT6 Trainingshistorie ... 77
5.3.7 FT7 Responsive Web-App & Querschnitt ... 78
6 Konzeption ... 83
6.1 Datenmodell ... 83
6.1.1 ERM ... 84
6.1.2 Löschverhalten ... 84
6.1.3 Umsetzung der Löschung ... 85
6.1.4 Eindeutigkeit ... 86
6.1.5 Prüfregeln ... 86
6.1.6 Indizes ... 86
6.1.7 Stammdatenwerte ... 87
6.2 Backend-Struktur ... 87
6.2.1 Klassendiagramme ... 88
6.2.1.1 Paketdiagramm ... 88
6.2.1.2 Benutzerverwaltung ... 88
6.2.1.3 Übungskatalog ... 89
6.2.1.4 Trainingsroutinen ... 89
6.2.1.5 Trainingsprotokoll Controller und Service ... 90
6.2.1.6 Trainingsprotokoll DTOs ... 90
6.2.1.7 Trainingsprotokoll Persistenz ... 90
6.2.1.8 Querschnitt ... 91
6.2.2 Business Regeln ... 91
6.2.2.1 Regeln im Service ... 91
6.2.2.2 Statusmodell der Trainingseinheit ... 92
6.3 REST-Schnittstelle ... 93
6.3.1 Grundsätze ... 93
6.3.2 Statuscodes ... 94
6.3.3 Endpunkte ... 94
6.3.4 Datenübertragungsobjekte ... 97
6.3.4.1 Request-DTOs ... 97
6.3.4.2 Response-DTOs ... 99
6.3.5 Einheitliches Fehlerformat ... 100
6.4 Abläufe ... 101

6.4.1 Aktivitätsdiagramme ... 101
6.4.1.1 SAF7 Training einplanen ... 101
6.4.1.2 SAF8 Training durchführen und protokollieren ... 102
6.4.2 Sequenzdiagramme ... 103
6.4.2.1 Login ... 103
6.4.2.2 Anfrage mit Token ... 104
6.4.3 Zustandsdiagramm Status Trainingseinheit ... 105
6.5 GUI-Konzept ... 105
6.5.1 Navigation ... 105
6.5.2 Screens ... 105
6.5.3 Wichtige Elemente ... 109
7 Entwicklung ... 110
7.1 Setup & Infrastruktur (Lokal) ... 110
7.2 Backend ... 110
7.2.1 Spring Boot ... 110
7.2.2 Vorgehen nach TDD ... 111
7.2.3 Testfälle ... 111
7.2.3.1 Unit-Tests Backend ... 112
7.2.3.2 Manuelle Systemtests Backend ... 114
7.2.4 Herausforderungen und Lösungen ... 114
7.2.5 README ... 115
7.2.6 Voraussetzungen ... 115
7.2.7 Umgebungsvariablen ... 115
7.2.8 Verwendete Technologien ... 116
7.2.9 Lokales Setup & Installation ... 116
7.2.9.1 Schritt 1: Repository klonen ... 116
7.2.9.2 Schritt 2: Datenbank-Setup ... 116
7.2.9.3 Schritt 3: Anwendung starten ... 117
7.2.10 Technische Details ... 117
7.2.10.1 Konfiguration ... 117
7.2.10.2 Dockerfile ... 117
7.2.10.3 Flyway-Migration ... 117
7.2.10.4 Spring Security ... 117
7.2.10.5 Transaction Handling ... 119
7.2.10.6 Exception Handling ... 119
7.2.10.7 Swagger ... 119

7.2.10.8 API-Versionierung 120
7.2.10.9 Actuator 120
7.3 Frontend 121
7.3.1 Manuelle Systemtests 121
7.3.1.1 Status Trainingseinheit 123
7.3.1.2 Löschkonzept 124
7.3.2 Unit-Tests 126
7.3.3 README 126
7.3.4 Voraussetzungen 126
7.3.5 Verwendete Technologien 127
7.3.6 Lokales Setup & Installation 127
7.3.6.1 Schritt 1: Repository klonen 127
7.3.6.2 Schritt 2: Abhängigkeiten installieren 127
7.3.6.3 Schritt 3: Backend starten 128
7.3.6.4 Schritt 4: Anwendung starten 128
7.3.6.5 Weitere Befehle 128
7.3.7 Technische Details 128
7.3.7.1 Ordnerstruktur 128
7.3.7.2 Vite-Proxy 129
7.3.7.3 Material UI 129
7.3.7.4 Fehlermeldungen 130
7.3.7.5 Dockerfile 130
7.3.8 Screenshots 130
7.3.8.1 Desktop 130
7.3.8.2 Mobile 133
8 Deployment 134
8.1 Reverse Proxy 134
8.2 Docker Compose 134
8.2.1 Reverse Proxy 134
8.2.2 Frontend 135
8.2.3 Backend 135
8.2.4 Datenbank 135
8.2.5 Logging 135
8.3 Deploy Workflow 135
8.4 Smoke-Tests 136
9 Benutzeranleitung 138

9.1 Überblick 138
9.2 Registrieren und anmelden 138
9.3 Übungen erfassen 138
9.4 Routinen zusammenstellen 138
9.5 Training planen 138
9.6 Training durchführen 139
9.7 Verlauf ansehen 139
9.8 Konto verwalten 139
10 Abschluss 140
10.1 Testergebnisse 140
10.2 Erfolgskontrolle 140
10.3 Projektrisikomanagement 142
10.4 Feedback Auftraggeber 142
10.5 Reflexion 142
10.6 Lessons learnt 143
10.7 Dankeschreiben 144
10.8 Kommentar zu KI 144
11 Anhang 145
11.1 Eigenständigkeitserklärung 145
11.2 Glossar 145
11.3 Quellenverzeichnis 152
11.4 Tabellenverzeichnis 153
11.5 Bildverzeichnis 155
11.6 Vorzeigetermine 157
11.6.1 Termin 1 157
11.6.2 Termin 2 158

1 Management Summary

Es wird eine digitale Plattform zur Trainingsverwaltung selbst entwickelt. Das Problem mit den öffentlich zugänglichen Lösungen ist, dass die guten Funktionen entweder hinter einer Paywall liegen oder die Open-Source-Apps unhandlich sind. Auf der Suche nach der perfekten Plattform wechselt man immer wieder den Anbieter und verliert dabei seine Trainingsdaten. Der ganze Trainingszyklus von der Planung bis zur Ausführung soll digital abgebildet werden. Als absolute Muss-Ziele dieser Diplomarbeit gelten eine Benutzerverwaltung, Übungen, Routinen, Planung, Live-Protokoll, Verlauf sowie ein responsive Frontend für Smartphones, Tablets und grössere Bildschirme.

Als Projektmanagementmethode wurde HERMES gewählt. Die Phasen bilden den groben Rahmen der Makroplanung. Kombiniert mit einem Kanbanboard für die tägliche Arbeit kann übersichtlich voran gearbeitet werden. Die Grundlage wurde durch ausführliche Technologieentscheidungen vorgegeben. Zusammen mit den Anforderungen entstand ein Konzept, welches als Bauplan für die Entwicklung gebraucht wurde.

Es gibt jetzt auf training.lennardbernet.ch eine komplette Plattform für die persönliche Trainingsverwaltung. Umgesetzt als Web-App ist sie öffentlich über HTTPS erreichbar. Nach der Registrierung landet man direkt angemeldet auf der Startseite. Man fängt an mit dem Erfassen von Übungen, erstellt eine Routine mit den Übungen und innerhalb weniger Minuten hat man seine wichtigsten Trainings eingeplant für die nächsten Tage. Das effektive Training kann in einer übersichtlichen Ansicht live protokolliert werden. Anschliessend kann über den Verlauf immer eingesehen werden, was man in den letzten Einheiten geleistet hat.

Die Qualität der Entwicklung wird durch automatisierte Tests sichergestellt. Mit einem automatischen Deployment und klar definierten Smoke-Tests ist die Grundlage für eine saubere Weiterentwicklung gelegt. Die eigenen Kontodaten sind gut geschützt. Es ist nicht möglich, die Daten eines anderen Benutzers zu sehen. Es wurden die wichtigsten Cyber Security Themen in Betracht gezogen und die nötigen Massnahmen umgesetzt.

Finanziell lohnt sich die eigene Entwicklung so einer Plattform definitiv nicht. Man muss mit einem fiktiven Betrag von rund 25'000 CHF rechnen für die Entwicklung und etwa 150 CHF Betrieb pro Jahr. Der Nutzen liegt in der Hoheit über die eigenen Daten und im persönlichen Kompetenzausbau als Softwareentwickler.

Der Zeitplan geriet ab September in Verzug, konnte aber rechtzeitig wieder aufgeholt werden. Das Projektrisikomanagement hat sich bewährt gegen die aufgetretenen Risiken. Schlussendlich wurden alle Muss-Ziele klar erfüllt. Die Kann-Ziele wurden bewusst nicht umgesetzt im Rahmen dieser Diplomarbeit. Trotzdem ist damit zu rechnen, dass die Entwicklung der Plattform weitergeht. Der Fokus wird auf einem Medien-Upload und Verlaufscharts liegen. Die Architektur ist bestens dafür vorbereitet.

2 Initialisierung

Die erste Phase dieses Projekts ist die Initialisierungsphase. Ziel der Initialisierung ist es den Projektumfang zu definieren, Projektmethodik festzulegen, Projektrisiken zu analysieren, eine Stakeholder Analyse durchzuführen und die Wirtschaftlichkeit zu prüfen. Diese Phase bildet das Fundament der Umsetzung.

2.1 Projektbeschrieb

2.1.1 Ausgangslage

Als Sportler organisiert man gerne seine Trainings. Ein beliebtes Tool dafür ist eine digitale Plattform, die man auf seinem Handy benutzen kann. Von diesen Apps oder Websites gibt es aktuell schon eine ganz grosse Palette. Erst wenn man sich mit den bestehenden Apps auseinandersetzt, fällt einem auf, dass die wirklich spannenden Features hinter einer Paywall versteckt sind. Wenn man auf die gratis oder Open-Source-Lösungen schaut, ist das Problem, dass sie entweder unhandlich zu bedienen sind oder schlichtweg für progressives Krafttraining die nötigen Features nicht besitzen.

Ich persönlich betreibe schon seit mehreren Jahren regelmässig Krafttraining und habe schon verschiedenste Apps hinter mir. Bis jetzt konnte ich aber noch nicht die perfekte App finden, ohne etwas dafür zu bezahlen. Es ist recht mühsam, wenn man immer wieder die Plattform wechseln muss, da man nach ein paar Wochen erst merkt, dass doch ein wichtiges Feature fehlt. Man verliert dabei bei einem Wechsel auch immer seine ganzen historischen Daten. Da ich im Rahmen dieses Studiums die nötigen Skills gelernt habe, um so eine Plattform selbst zu entwickeln, die genau auf meine Bedürfnisse zugeschnitten ist, packe ich diese Chance. Meine Motivation, dieses Produkt schlussendlich auch selbst zu benutzen, treibt mich an, um hier saubere Arbeit zu leisten.

In dieser Diplomarbeit wird also eine Applikation entwickelt, welche eine moderne intuitive Bedienung hat, wie die kommerziellen Apps, aber die Möglichkeiten bietet, welche die aktuellen Open-Source Anbieter haben. Durch eine browserbasierte Lösung ist das Endprodukt plattformunabhängig.

2.1.2 Projektumfang und Zieldefinition

Der Rahmen dieser Diplomarbeit wird klar definiert. Somit ist der Projekterfolg messbar und man vermeidet einen Scope-Creep. Dazu werden die verschiedensten Funktionen in Muss und Kann-Ziele unterteilt. Ebenfalls werden Aspekte, die nicht Teil dieser Arbeit sind, erwähnt.

Muss-Ziele

Diese Ziele bieten die Grundlage der Software und müssen eingehalten werden für eine erfolgreiche Diplomarbeit.

- Implementierung einer vollumfänglichen Benutzerverwaltung inklusive Registrierung und Login serverseitig. Dazu gehört auch ein Löschkonzept, bei dem Nutzer ihre Daten komplett entfernen können.
- Verwaltungsfunktionen, um den Übungskatalog und Trainingsroutinen individuell zu erstellen, bearbeiten und löschen.
- Möglichkeit, um eine Trainingsroutine für zukünftige Daten einzuplanen.

- Einsicht auf die Historie einer Trainingsroutine.
- Eine Trainingseinheit live protokollieren mit dem Erfassen von Metadaten der Sätze, Wiederholungen, Gewichten, Pausenzeiten sowie weitere Informationen wie Muskelgruppe festlegen und eine Beschreibung hinzufügen.
- Umsetzung als vollständige responsive Web-Applikation wodurch eine optimale Bedienung auf dem Smartphone sowie einem Desktop gewährleistet ist.

Kann-Ziele

Diese Features sind optional und werden erst angegangen, wenn die Muss-Ziele zu 100% abgeschlossen sind.

- Integration eines Medien-Uploads, um Fotos an spezifischen Übungen oder Trainingseinheiten anzuhängen.
- Visuelle Aufbereitung der Trainingseinheiten in der Form von Verlaufscharts, Tabellen, Listen oder sogar grafische Dashboards.
- Verschiedene Rollen (Trainer oder Sportler) mit einer Kundenverwaltung.

Nicht-Ziele

Für die klare Abgrenzung der Diplomarbeit werden folgende Nicht-Ziele definiert.

- Keine native Mobile App und somit auch keine Veröffentlichung in kommerziellen App-Stores.
- Keine Integration von Wearables oder Smartwatches.
- Keine Echtzeit Funktionen wie zum Beispiel ein Live-Chat oder Überwachung zwischen Trainer und Sportler.

2.1.3 Projektvision

Die Vision dieses Projekts ist es den gesamten Zyklus eines Krafttrainings von der Planung bis zur aktiven Ausführung komplett digital abzubilden und zu verwalten. Anstatt sich mit unübersichtlichen Menüs oder Paywalls herumzuschlagen zu müssen, hat man als Nutzer eine Plattform mit uneingeschränktem Zugriff auf die wichtigsten Funktionen für das progressive Krafttraining.

Im Zentrum steht dabei klar eben dieser reibungslose Ablauf des gesamten Trainingszyklus. Wenn man im Fitnessstudio ist, möchte man sich nicht auf der Oberfläche der App verlieren, sondern genau wissen, wo welche Features was machen. Somit kann man ein sauberes Training planen und sobald man im Studio ist, rückt die Technik in den Hintergrund und die Applikation dient rein der Unterstützung. Durch die solide Architektur kann garantiert werden, dass die wertvollen Trainingsdaten jeweils präzise, langfristig und sicher verarbeitet werden. Der Nutzer hat jederzeit die absolute Kontrolle und kann sein Profil und die Daten komplett selbst verwalten.

Durch die Kombination aus technischer Verlässlichkeit und funktionaler Bedienbarkeit ist die Basis für ein zukunftssicheres Werkzeug gelegt. Erweiterungen wie visuelle Aufbereitung der Trainingshistorie oder sogar Integration von Wearables sind langfristig möglich. Es soll die Freiheit von Open-Source Lösungen mit dem Bedienkomfort einer kommerziellen Software verknüpft werden.

2.2 Projektmanagement

Für die Umsetzung dieses Projekts wird ein hybrider Ansatz aus HERMES und Kanban gewählt.

2.2.1 Makro-Management nach Hermes

Für die übergreifende Steuerung des Projekts wird die Projektmanagement-Methode HERMES angewendet. Sie eignet sich für IT-Projekte, da sie eine klare Struktur mit definierten Phasen und Meilensteinen vorgibt. Offiziell gibt es die Phasen Initialisierung, Konzept, Realisierung und Einführung [1]. Während dieser Diplomarbeit werden diese Phasen etwas erweitert. Das Projekt wird in folgende klar definierte, übergreifende Phasen unterteilt:

- Initialisierung: Festlegung der Ausgangslage, Ziele und Risiken.
- Planung: Definieren der Meilensteine und grober Zeitplan.
- Vorbereitung: Recherche der technischen Möglichkeiten.
- Anforderungsanalyse: Die jeweiligen Anforderungen klar ausarbeiten.
- Konzeption: Modellierung der Datenstruktur, Klassendiagramme, Aktivitätsdiagramme, Grobdesign der Benutzeroberfläche
- Entwicklung & Testing: Implementierung anhand der Konzepte mit kontinuierlichen Tests, um die Qualitätssicherung sicherzustellen
- Deployment: Aufbau und Einrichtung einer automatisierten CI/CD-Pipeline
- Abschluss: Fertigstellung der Dokumentation, Reflexion und Schlusswort

2.2.2 Mikro-Management mit Kanban

Für die effektive Umsetzung dieser Projektphasen wird Kanban angewendet. Dies ist eine Projektmanagement-Methode, womit man die Arbeit visualisieren kann mit einem Kanban Board. Alle offenen Aufgaben werden auf einem digitalen Kanban-Board erfasst und durchlaufen die Spalten «ToDo», «In Bearbeitung» und «Erledigt». Diese Aufgaben haben jeweils auch Abhängigkeiten untereinander. Anhand dieser Abhängigkeiten kann priorisiert werden, woran man arbeitet. Um Engpässe zu vermeiden, wird ein Work in Progress Limit (WIP-Limit) festgelegt von 2. Jede Aufgabe erhält eine «Definition of Done». Somit ist im Vorfeld klar definiert, was die Erfolgskriterien sind, um eine Aufgabe abzuschliessen [2].

Das Kanbanboard wird initialisiert ab der Projektphase «Planung». Es soll jeweils pro Phase als erstes die erwarteten Aufgaben auf dem Kanbanboard hinzugefügt werden.

2.3 Projektrisikomanagement

Mit einem ausgearbeiteten Projektrisikomanagement hat man die Werkzeuge für die Erkennung, Analyse und Bewältigung potenzieller Gefahren, die das Projekt beeinträchtigen könnten. Kein Projekt läuft komplett so wie es ursprünglich geplant wurde. Was zählt, ist, dass man auf diese unerwarteten Herausforderungen gut vorbereitet ist. Der korrekte Umgang mit den Risiken im Projektmanagement gibt eine solide Grundlage, um Strategien zu entwickeln welche das Projekt auf dem Kurs halten. Diese Risiken können während des gesamten Lebenszyklus des Projekts auftreten [3].

2.3.1 Risikoerkennung

Als Einzelentwickler für eine komplette Full-Stack Web-Applikation inklusive Deployment hat man ganz andere Herausforderungen als ein grosses Team. Die Risiken wurden anhand von Erfahrungen aus früheren Projekten evaluiert.

Technische Risiken

- **Unterschätzung des Frontend-Aufwands:** Die Umsetzung des Frontends (React/CSS) wird zeitlich unterschätzt, da der Kern-Fokus der eigenen Expertise eher auf dem Backend liegt.
- **Verzögerungen beim Deployment-Setup:** Zeitverlust beim Aufsetzen der automatisierten CI/CD Pipeline mit Docker, da hier die Vorkenntnisse nicht sehr ausgeprägt sind.
- **Over-Engineering der Backend-Architektur:** Eine hohe Detailliebe und der Fokus auf komplexe Backend-Strukturen benötigen viel Entwicklungszeit.
- **Responsive Testing & Mobile-Darstellung:** Gerätespezifische Bugs, Touch-Probleme und Layout-Fehler auf den physischen Smartphones fallen erst sehr spät auf, da primär am Desktop-Monitor programmiert wird.
- **Komplexität der Security-Konzepte:** Fehler bei der Implementierung von Security-Konzepten können zu Schwachstellen führen.
- **Breaking Changes in Abhängigkeiten:** Major-Releases von eingesetzten Drittbibliotheken können zu unerwartetem Refactoring-Aufwand führen in der Entwicklungsphase.

Organisatorische Risiken

- **Scope Creep:** Wenn man zu früh mit den Kann-Zielen anfängt, kann das zu Verzögerungen bei der Fertigstellung der Muss-Ziele führen.
- **Engpass als Einzelentwickler:** Da nur eine Person an diesem Projekt arbeitet übernimmt diese die Rolle als Projektmanager, Entwickler und Tester. Das kann bei Problemen zu einem Flaschenhals führen.
- **Verfügbarkeit:** Zeitliche Engpässe für die Diplomarbeit durch unerwartete Belastungen im Privatleben oder Beruf.
- **Bias beim Testing:** Da nur eine Person programmiert und testet kann das dazu führen, dass gewisse Workflows im Testing gar nicht erkannt werden.

2.3.2 Risikoanalyse

Die identifizierten Risiken müssen genauer analysiert werden, bevor man eine Priorisierung dazu machen kann. Dafür wird eine Risikomatrix erstellt. Diese Bewertung in der Matrix erfolgt über zwei Achsen:

- **Eintrittswahrscheinlichkeit:** (1 bis 6: 1 = Gering, 3 = Mittel, 6 = Hoch)
- **Schadenausmass:** Wie stark gefährdet das Risiko den Projekterfolg? (1 bis 6)
- **Risikowert:** Multiplikation aus beiden Werten (Wahrscheinlichkeit x Ausmass). Das gibt direkt eine Priorisierung.

2.3.3 Risikomatrix

Mit den erkannten Risiken und vordefinierten Parametern für die Analyse dieser Risiken kann eine Matrix daraus erstellt werden.

Tabelle 1: Risikobewertung

ID	Risiko	Wahrscheinlichkeit (1-6)	Ausmass (1-6)	Risikowert
R01	Unterschätzung des Frontend-Aufwands	4	5	20
R02	Verzögerungen beim Deployment-Setup	4	4	16
R03	Over-Engineering der Backend-Architektur	5	5	25
R04	Responsive Testing & Mobile-Darstellung	4	3	12
R05	Komplexität der Security-Konzepte	3	5	15
R06	Breaking Changes in Abhängigkeiten	1	4	4
R07	Scope Creep	2	4	8
R08	Engpass als Einzelentwickler	6	4	24
R09	Verfügbarkeit	3	4	12
R10	Bias beim Testing	5	4	20

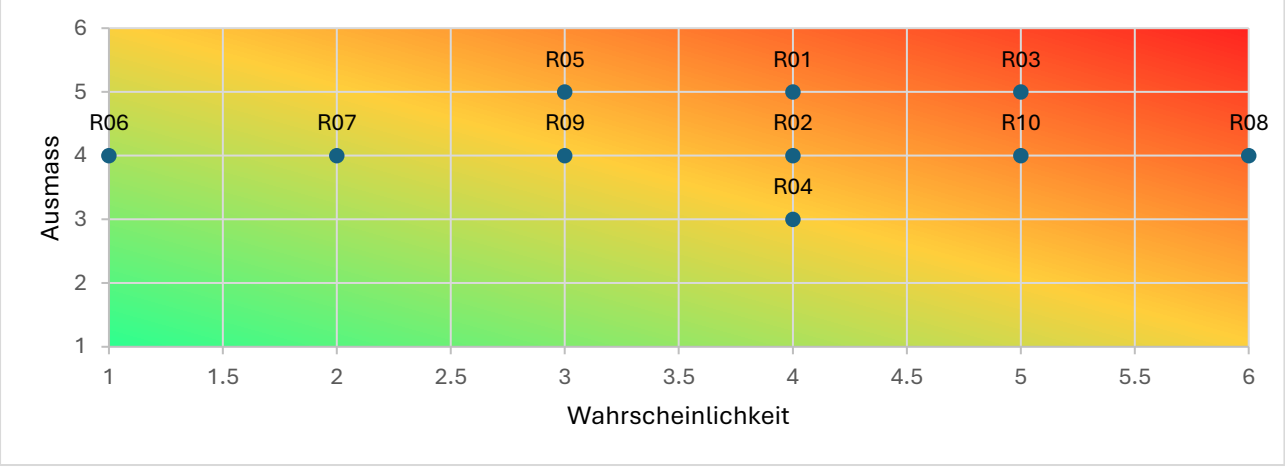


Abbildung 1: Risikomatrix

2.3.4 Massnahmen

Bei den erkannten Risiken kann man effektive Massnahmen ergreifen um das Ausmass so früh wie möglich zu minimieren. Nach der Bewertung des Risikos kann man jeweils eine Strategie für den Umgang des Risikos festlegen. Jedem Risiko wird eine der folgenden vier Präventionsstrategien zugewiesen: [4]

1. **Vermeidung:** Die Ursache des Risikos wird komplett eliminiert, indem man auf gewisse Technologien oder Funktionen verzichtet.
2. **Verminderung:** Die Eintrittswahrscheinlichkeit oder das Schadenausmass werden durch gezielte Massnahmen reduziert.
3. **Auslagerung:** Das Risiko wird an Dritte weitergegeben.
4. **Akzeptanz:** Man nimmt das Risiko bewusst in Kauf, falls es eintritt, aber man hat einen Notfallplan definiert.

Massnahmen für R03 & R08 (Over-Engineering & Einzelentwickler – Risikowert 24/25)

Strategie: Verminderung

Die Backend-Tasks haben eine sehr klare «Definition of Done». Somit wird an der Architektur des jeweiligen Moduls nicht weitergearbeitet, obwohl es eigentlich schon funktioniert. Der Fokus bleibt beim Wesentlichen und nicht auf Details.

Das definierten WIP-Limit der Tasks auf dem Kanbanboard muss rigoros eingehalten werden. Ansonsten läuft man in die Gefahr, dass man die Muss-Ziele nicht mehr im Fokus hat.

Massnahmen für R01 (Frontend-Aufwand – Risikowert 20)

Strategie: Vermeidung

Der Fokus wird auf dem Backend liegen, da dort die eigenen Kompetenzen am stärksten sind. Im Frontend sollten aufwendige Custom CSS Ansätze vermieden werden mit dem Einsatz von etablierten Komponentenbibliotheken. Keine grossen Experimente starten.

Massnahmen für R10 (Bias beim Testing – Risikowert 20)

Strategie: Verminderung

Da kein externer Tester eingebunden wird, ist der Fokus auf automatisierte Tests wichtig. Die erwarteten Edge Cases sollen früh erkannt werden und in das Testkonzept aufgenommen werden. Es soll ein Test Driven Development (TDD) Ansatz angewendet werden, um die Fehlerquoten zu minimieren.

Massnahmen für R02 (Deployment-Setup – Risikowert 16)

Strategie: Verminderung

Das ganze Deployment soll nicht komplett bis zur finalen Phase aufgeschoben werden. Schon während einer früheren Entwicklungsphase muss ein initialer «Proof of Concept» für den Docker-Container aufgesetzt werden. Somit läuft man nicht ins Risiko, dass man in der Endphase einen grossen Druck verspürt, da noch gar nichts vom Deployment erledigt wurde.

Massnahmen für R05 (Security-Konzepte – Risikowert 15)

Strategie: Auslagerung

Die Security Konzepte werden von branchenüblichen und etablierten Frameworks übernommen. Die Implementierung von JWT oder ein Löschkonzept erfolgen nach den Best Practices ohne dass man komplizierte Verschlüsselung oder Session-Mechanismen erfindet.

Massnahmen für R04 (Responsive Testing & Mobile-Darstellung – Risikowert 12)

Strategie: Verminderung

Um nicht ganz am Schluss das ganze Frontend neu zu designen, wird von Anfang an kontinuierlich auch das Responsive Design in der Entwicklung berücksichtigt. Man sollte auch versuchen, die Applikation direkt auf dem physischen Smartphone aufzurufen, um die Touch-Interaktionen und das Layout unter realen Bedingungen zu testen.

Massnahmen für R09 (Verfügbarkeit – Risikowert 12)

Strategie: Akzeptanz

Unerwartete private oder berufliche Belastungen können nicht vollständig kontrolliert oder vermieden werden. Zur Abfederung dieses Risikos kann man während der Planung bewusste Puffer zwischen den Meilensteinen einplanen.

Tritt ein massiver zeitlicher Engpass auf, wird die Entwicklung kompromisslos auf die festgelegten Muss-Ziele reduziert.

Massnahmen für R07 (Scope Creep – Risikowert 8)

Strategie: Vermeidung

Wenn man während dem Projekt die Disziplin bewahrt und man nur an Funktionen arbeitet, welche auch in den Anforderungen spezifiziert sind, spart man viele Ressourcen. Auf dem Kanban-Board werden die Kann-Ziele so lange wie möglich zurückgehalten.

Massnahmen für R06 (Breaking Changes in Abhängigkeiten – Risikowert 4)

Strategie: Verminderung

Während der Hauptentwicklungsphase werden die gewünschten Versionen der Abhängigkeiten festgelegt. Somit läuft man nicht in einen unnötigen Refactoring-Aufwand bei Breaking Changes. Nur wenn kritische Sicherheitslücken auftreten, kann man auf eine neuere Version wechseln.

2.4 Stakeholder Analyse

Es werden die Stakeholder dieser Diplomarbeit identifiziert. Die Wichtigkeit dieser Stakeholder wird anhand des Risikos und Aufwand gemessen. Somit findet man heraus, wer denn überhaupt Anforderungen an diese Diplomarbeit stellen könnte. Es wird versucht alle beteiligten Parteien herauszufinden, die direkt oder indirekt betroffen sind.

Das Risiko bestimmt jeweils, wie gross die Auswirkung ist, wenn dieser Stakeholder nicht berücksichtigt wird. (1 = kein, 2 = gering, 3 = mittel, 4 = hoch, 5 = sehr hoch, 6 = fatal)

Der Aufwand sagt aus wie viel Zeit man investieren muss, um diesem Stakeholder zu berücksichtigen. (1 = extrem, 2 = sehr hoch, 3 = hoch, 4 = mittel, 5 = gering, 6 = vernachlässigbar)

2.4.1 Stakeholder

2.4.1.1 Entwickler

Als alleiniger Entwickler und schlussendlich auch Endanwender bin ich für die komplette Umsetzung zuständig. Gleichzeitig bin ich als Sportler auch der einzige Systembetroffene und Fachexperte für das progressive Krafttraining.

Risiko (6 – fatal)

Falle ich als Einzelentwickler aus oder verliere die Motivation, wird nichts mehr an dieser Diplomarbeit gemacht und das ganze Projekt steht still. Ebenfalls verfehlt das Projekt die Vision, wenn die Applikation nicht meine Anforderungen an das Tracking des Krafttrainings erfüllt.

Aufwand (6 – vernachlässigbar)

Da ich Einzelentwickler bin gibt es keine internen Kommunikationswege wie in einem Team. Abstimmungen oder Missverständnisse sind daher kein Problem.

2.4.1.2 Diplomcoach & Diplomexperte

Dominic Merz (Diplomcoach) und Milad Fakiry (Diplomexperte)

Der Diplomcoach und Experte agieren als Anforderungsverantwortliche dieser Diplomarbeit. Sie sind befugt, um die Qualität der Diplomarbeit final zu bewerten.

Risiko (5 - sehr hoch)

Wenn die Erwartungen an der Diplomarbeit, die Methodik oder sonst etwas an der Dokumentation nicht erfüllt werden, ist das erfolgreiche Bestehen der Diplomarbeit gefährdet.

Aufwand (3 – hoch)

Durch das Einbeziehen des Diplomcoachs kann sichergestellt werden, dass man zielgerichtet an der Diplomarbeit arbeitet. Daher macht es durchaus Sinn hier Zeit zu investieren. Die Erfahrung und Feedbacks des Diplomcoachs sind viel wert für die finale Bewertung.

2.4.1.3 Schule

Die Schule (TEKO) kann man in diesem Projekt als «Gesetzgeber» anschauen. Sie definiert die Rahmenbedingungen und Spielregeln dieser Diplomarbeit. Alle Vorgaben werden von der Schule festgelegt.

Risiko (4 – hoch)

Es müssen die Vorgaben der Schule eingehalten werden. Wird dies nicht gemacht kann das ganz schnell zum Nichtbestehen des Projekts führen, unabhängig von der Softwarequalität.

Aufwand (5 – gering)

Die Richtlinien sind klar vorgegeben und daran lässt sich nichts ändern. Es wird transparent kommuniziert von der Schule, weshalb keine aktive Kommunikation nötig ist. Dafür ist jeweils der Diplomcoach zuständig.

2.4.1.4 Hosting und Infrastruktur Provider

Hierbei handelt es sich um externe Service-Dienstleister welche die Infrastruktur für das Deployment (Docker, Datenbank, Domain) bereitstellen.

Risiko (3 – mittel)

Eine genaue Analyse des Providers ist notwendig um technische Ausfälle, Limitierungen oder Probleme beim Setup zu vermindern.

Aufwand (5 – gering)

Im Optimalfall hat man wenig bis keinen Austausch mit dem Provider, da die Infrastruktur meistens einfach funktioniert.

2.4.2 Bewertung der Stakeholder

Um die Relevanz der einzelnen Stakeholder abschliessend zu bewerten, wird für jeden ein Prioritätswert berechnet. Diese Berechnung erfolgt mit folgender Formel:

Priorität = Wurzel aus (Risiko^2 + Aufwand^2)

Anhand der berechneten Priorität lassen sich die Stakeholder in drei Prioritätsgruppen unterteilen:

Muss (5.0 – 8.5): Unbedingt berücksichtigen. Ohne diese Stakeholder kann das ganze Projekt sehr schnell schieflaufen.

Sollte (3.4 – 4.9): Nach Möglichkeit berücksichtigen. Haben keine hohe Priorität.

Könnte (1.0 – 3.3): Diese Stakeholder kann man am ehesten vernachlässigen.

Tabelle 2: Bewertung der Stakeholder

Stakeholder	Risiko	Aufwand	Priorität	Kategorie
Entwickler	6	6	8.5	Muss
Schule	4	5	6.4	Muss
Diplomcoach & Diplomexperte	5	3	5.8	Muss
Hosting und Infrastruktur Provider	3	5	5.8	Muss

Wie man sieht, fallen alle Stakeholder für diese Diplomarbeit unter die Kategorie «Muss». Dies liegt auch an der Natur des Projekts, da es eine Einzelarbeit ist. Trotzdem ist eine Priorisierung zu erkennen.

Rein theoretisch können auch weitere Benutzer die App benutzen, da sie öffentlich ist. Diese werden nicht als Stakeholder aufgezeigt, da sie dieselben, oder weniger Bedürfnisse haben wie der Entwickler.

2.5 Wirtschaftlichkeitsanalyse

Das Ziel dieser Analyse ist es, herauszufinden, wie sich die Eigenentwicklung langfristig gegenüber bestehenden kostenpflichtigen Marktangeboten verhält.

Dafür muss erst mal klar definiert werden, welche zwei Wege man gehen kann.

2.5.1 Buy-Case

Für die Übersicht der aktuellen Marktsituation wurde mithilfe von Google Gemini die beliebtesten Apps für das Krafttraining sowie was hinter der Paywall ist, ermittelt. Dabei wurden folgende 3 Apps als Ergebnis geliefert: [5]

Tabelle 3: Vergleich bestehender Apps

App	Strong	Hevy	Fitbod
Beschreibung	Gilt als absoluter Klassiker in der Fitness-Community, mit einem klaren Fokus auf ein simples und schnelles Interface.	Ein sehr moderner Konkurrent, der sich durch eine saubere Benutzeroberfläche, integrierte soziale Features und vor allem durch eine vollständige und flüssige Android-Kompatibilität auszeichnet.	Richtet sich eher an Nutzer, die weniger selbst planen, sondern sich ihr Training von einem Algorithmus basierend auf vorhandenem Equipment und Muskelregeneration generieren lassen wollen.
Was wird blockiert von der Paywall	Beschränkung der Trainingsroutinen. Kein Datenexport. Erweiterte Tracking-Tools.	Anzahl der Accounts, Routinen und Übungen. Keine grafische Auswertung der Trainingshistorie.	Die App sperrt sich komplett nach einer 7-tägigen Testphase.
Kosten für uneingeschränkte Nutzung	30$ jährlich	Zwischen 24$ und 40$ jährlich. Alternativ eine Lifetime-Lizenz für rund 75$	95$ jährlich

2.5.2 Make-Case

Der andere Ansatz ist eben die Eigenentwicklung trotz des hohen Aufwands.

Hier ist eine Liste mit Argumenten, die für die Eigenentwicklung sprechen:

- **Absolute Datenhoheit:** Die Daten und Auswertungen gehören zu 100% einem selbst.
- **Massgeschneiderte Features:** Die Anwendung deckt die exakt definierten Muss-Ziele ab, die für ein progressives Krafttraining wirklich zählen. Unnötige Features werden ignoriert.
- **Volle Plattformunabhängigkeit:** Durch die Web-Applikation kann das Tool auf allen Endgeräten funktionieren. Egal ob man am Desktop-Monitor oder einem Handygerät ist.
- **Persönliche Entwicklung:** Dieses Projekt ist ausschlaggebend für den erfolgreichen Abschluss als Dipl. Informatiker HF.

- **Gezielter Kompetenzaufbau:** Das Projekt kann genauso gesteuert werden, dass die persönlichen Stärken genutzt werden. Gleichzeitig baut man in wichtigen Bereichen der Softwareentwicklung weiteres Know-how auf.
- **Hohe Motivation:** Da der Entwickler gleichzeitig auch der Endanwender ist, besteht ein grosser Eigenantrieb, um durchdachte Konzepte und eine hohe Code-Qualität sicherzustellen.

Die kommerziellen Alternativen haben keine initiale Entwicklungszeit, bevor man die Tools gebrauchen kann. Jedoch gibt es trotzdem Restriktionen, welche man nicht komplett umgehen kann. Für den persönlichen langfristigen Nutzen liefert die Eigenentwicklung eine exakt zugeschnittene Software und volle Datenkontrolle. Ebenfalls dient diese Projektarbeit als entscheidender Faktor für den HF-Abschluss.

2.5.3 Opportunitätskosten

Da es sich um ein eigenes Privatprojekt handelt fliessen keine Lohnzahlungen direkt in die Entwicklung. Dennoch kann man die investierte Zeit als fiktive Investition mal durchrechnen.

Rechnet man für die initiale Umsetzung grob mit 250 Stunden und einem Stundensatz von 100 CHF, ergibt das einen Betrag von 25'000 CHF. Anschliessend hat man aber ein ausgearbeitetes Konzept, Fullstack Entwicklung sowie eine CI/CD-Pipeline. In der Planung kommt man schlussendlich sogar auf 302 Stunden was einen Betrag von 30'200 CHF enspricht

2.5.4 Betriebskosten

Falls man die fertige Applikation nach dem Abschlussdatum der Diplomarbeit weiter benutzen will, fallen wiederkehrende externe Kosten an. Da die Anbieter noch nicht klar sind werden hier Durchschnittswerte gebraucht.

Hosting & Server

Es wird eine Infrastruktur mit Frontend, Backend und Datenbank auf Docker-Containern betrieben. Für einen verlässlichen Service muss man mit ca. 10 bis 15 CHF pro Monat rechnen.

Domain & SSL

Da es sich um eine Web-Applikation handelt braucht man eine Domain, die über HTTPS erreichbar ist. Die Kosten für eine Standard-Domain belaufen sich auf ca. 15 CHF pro Jahr.

Somit kommt man auf ein Total von ca. 150 CHF pro Jahr. Aufwände für die Wartung und Updates nach der Diplomarbeit werden hier nicht mitgerechnet.

2.5.5 Fazit

Aus rein wirtschaftlicher Sicht lohnen sich die fiktiven Entwicklungskosten sowie die jährlichen Betriebskosten überhaupt nicht. Das Projekt ist eigentlich hochgradig unwirtschaftlich.

Wie schon erwähnt ist die Gewichtung aber massiv auf dem qualitativen Nutzen. Die Entwicklung hängt eng zusammen mit dem persönlichen Erwerb des HF-Diploms. Im Verlauf der Diplomarbeit werden viele Kompetenzen der Softwareentwicklung neu gelernt oder bestehendes Wissen verbessert. Das Endprodukt ist eine massgeschneiderte, plattformunabhängige Lösung, die genau auf das progressive Krafttraining zugeschnitten ist. Wenn man das berücksichtigt, ist es durchaus plausibel, dass diese Diplomarbeit so durchgeführt wird.

3 Planung

3.1 Makro Planung

Die im Projektmanagement aufgeführten Phasen werden hier genauer definiert. Die Anfangs- und Enddaten der einzelnen Phasen sind grobe Vorgaben und sollten während dem Projekt so weit eingehalten werden wie möglich. Am wichtigsten ist der Abgabetermin am 28. Oktober 2026, dieser muss zwingend eingehalten werden. Bis dann muss die letzte Phase «Abschluss» fertiggestellt werden.

Tabelle 4: Projektphasen

Phase	Startdatum	Enddatum	Ziel / Ergebnis
Initialisierung	12.6.2026	28.6.2026	In dieser Phase werden die Grundlagen des Projekts festgelegt. Festlegung der Ausgangslage, Ziele, Risiken, Stakeholder und Wirtschaftlichkeit.
Planung	28.6.2026	12.7.2026	Anhand der Projektmanagementmethode die Planung durchführen. Somit hat man eine detaillierte Übersicht der nächsten Phasen.
Vorbereitung	12.7.2026	26.7.2026	Diese Phase dient dazu, um anhand der Ausgangslage und Vorkenntnisse die ersten wichtigen Entscheidungen der Architektur zu treffen.
Anforderungs-analyse	26.7.2026	16.8.2026	Definieren der Geschäftsanwendungsfälle, Systemanwendungsfälle und daraus die konkreten Anforderungen ableiten.
Konzeption	16.8.2026	6.9.2026	Modellierung der Datenstruktur, Klassendiagramme, Aktivitätsdiagramme, Grobdesign der Benutzeroberfläche
Entwicklung & Testing	6.9.2026	4.10.2026	Effektive Entwicklungsphase in der man die Anforderungen und Konzeption umsetzt. Die Applikation muss ebenfalls vollständig getestet sein.
Deployment	4.10.2026	18.10.2026	Aufbau und Einrichtung einer automatisierten CI/CD-Pipeline
Abschluss	18.10.2026	28.10.2026	Fertigstellung der Dokumentation, Reflexion und Schlusswort

3.1.1 GANTT Diagramm



ID	Phase / Meilenstein	Soll Aufwand (h)	Soll Start	Soll Ende	Ist Start	Ist Ende
1	**Initialisierung**	24	12.06.26	28.06.26	12.06.26	24.06.26
2	**Planung**	14	28.06.26	12.07.26	27.06.26	05.07.26
M1	*Meilenstein 1 – Planung*		*12.07.26*	*12.07.26*	*12.07.26*	*12.07.26*
3	**Vorbereitung**	40	12.07.26	26.07.26	18.07.26	01.08.26
4	**Anforderungsanalyse**	30	26.07.26	16.08.26	02.08.26	23.08.26
5	**Konzeption**	42	16.08.26	06.09.26	05.09.26	05.10.26
M2	*Meilenstein 2 – Konzeption*		*06.09.26*	*06.09.26*	*05.10.26*	*05.10.26*
6	**Entwicklung & Testing**	94	06.09.26	04.10.26	08.10.26	19.10.26
7	**Deployment**	32	04.10.26	18.10.26	20.10.26	22.10.26
M3	*Meilenstein 3 – Deployment*		*18.10.26*	*18.10.26*	*22.10.26*	*22.10.26*
8	**Abschluss**	26	18.10.26	28.10.26	23.10.26	28.10.26
M4	*Meilenstein 4 – Abschluss (Abgabe)*		*28.10.26*	*28.10.26*	*28.10.26*	*28.10.26*
	Total Projekt	302	12.06.26	28.10.26		

Abbildung 2: GANTT Diagramm

3.1.2 Meilensteine

Meilenstein 1 – Planung

Mit dem Abschluss der Planung hat man eine ausgearbeitete Grundlage des Projekts. Man hat sichergestellt, dass das Projekt gut aufgegleist ist und man die wichtigsten Aspekte rechtzeitig berücksichtigt.

Meilenstein 2 – Konzeption

Anhand der Vorbereitung, Anforderungsanalyse und Konzeption hat man eine perfekte Vorlage für die Entwicklung. Es ist klar, welche Technologien benutzt werden und man kann anhand der Anforderungen die Tests schon grösstenteils vorab definieren. Somit sollte die Entwicklung grösstenteils eine reine Fleissarbeit sein.

Meilenstein 3 – Deployment

Die Software wurde komplett entwickelt und ist im Browser jederzeit verfügbar. Das Produkt ist abgeschlossen.

Meilenstein 4 – Abschluss

Das ganze Projekt ist umfangreich und genau dokumentiert. Die Abgabe kann nach Vorgabe durchgeführt werden.

3.2 Mikro Planung

3.2.1 Arbeitspakete

Der Aufwand dieser Arbeitspakete beinhaltet jeweils bewusst etwas mehr Stunden, als eigentlich erforderlich wäre. Dadurch hat man ausreichend Spielraum, um auf unvorhergesehene Aufwände zu reagieren, ohne dass die komplette Planung über den Haufen geworfen wird.

Die Abhängigkeiten innerhalb der Phasen sind einzuhalten. Erst wenn man alle Arbeitspakete einer Phase erledigt hat, kann man mit der nächsten beginnen.

Es wird pro Arbeitspaket eine Definition of Done vordefiniert. Diese kann sich auch im Verlauf des Projekts noch verändern. Ein Arbeitspaket darf erst auf «Erledigt» gesetzt werden, sobald diese Bedingung erfüllt wurde.

Diese Arbeitspakete werden so ins Jira übernommen für das Kanban Tool. Wie in den Massnahmen des Projektrisikomanagements festgelegt muss das WIP-Limit von 2 unbedingt eingehalten werden.

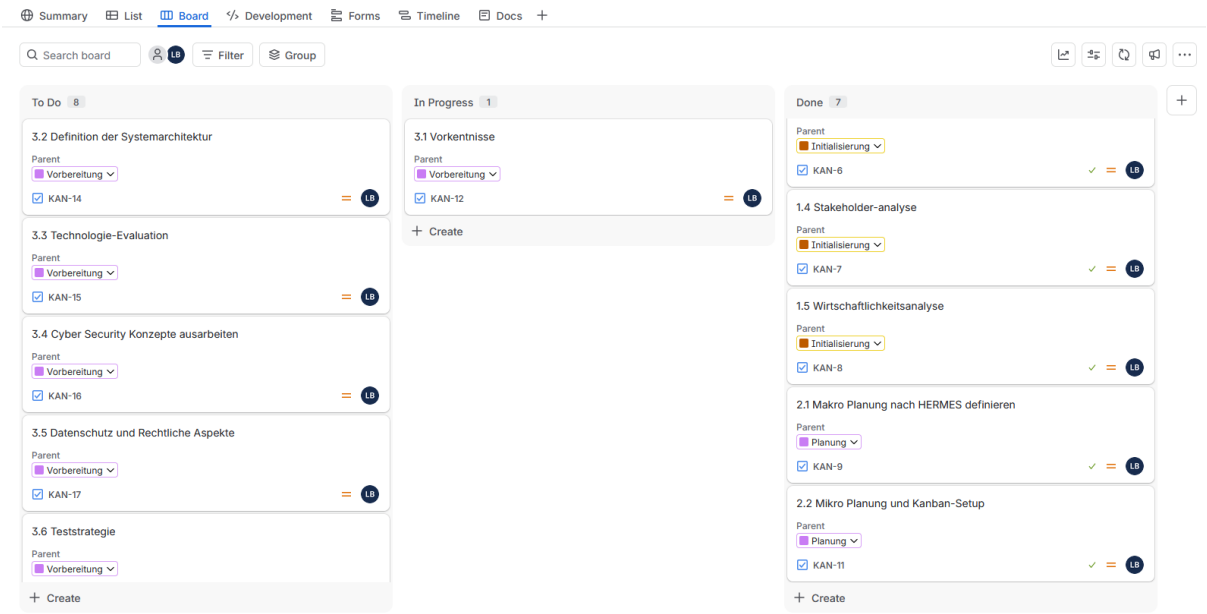


Abbildung 3: Kanban-Board in Jira

Die initialen Arbeitspakete werden mit blau markiert. Fällt während der Entwicklung auf, dass ein Zwischenschritt fehlt, man ein komplettes Arbeitspaket neu hinzufügen muss oder etwas entfernt wird, wird dies mit orange markiert.

3.2.1.1 Initialisierung

Tabelle 5: Arbeitspakete Initialisierung

ID	Name	Definition of Done	Aufwand	Abhängigkeit
1.1	Projektidee und Ausgangslage definieren	Das Kapitel «Projektbeschrieb» beinhaltet die Ausgangslage, Alle Ziele (Muss, Kann, Nicht) und eine persönliche Projektvision.	6 Stunden	Keine
1.2	Projekt-management	Der grobe Ablauf des Projekts ist anhand einer Projektmanagement-Strategie definiert.	4 Stunden	1.1
1.3	Projektrisikomana gement durchführen	Alle erkannten Risiken sind in der Matrix anhand von Eintrittswahrscheinlichkeit und Schadenausmass bewertet. Für jedes identifizierte Risiko ist eine konkrete Präventionsstrategie	6 Stunden	1.2

		(Vermeidung, Verminderung, Auslagerung oder Akzeptanz) und Massnahme definiert.		
1.4	Stakeholder-analyse	Die Akteure für den Projekterfolg sind identifiziert und bewertet.	4 Stunden	1.1
1.5	Wirtschaftlichkeits analyse	Die Wirtschaftlichkeit des Projekts wurde analysiert.	4 Stunden	1.1

3.2.1.2 Planung

Tabelle 6: Arbeitspakete Planung

ID	Name	Definition of Done	Aufwand	Abhängigkeit
2.1	Makro Planung nach HERMES definieren	Alle Projektphasen sind mit Startdatum und Enddatum definiert. Die Meilensteine wurden deklariert.	6 Stunden	Initialisierung beendet
2.2	Mikro Planung und Kanban-Setup	Definition der Arbeitspakete und diese im Kanbanboard einrichten.	8 Stunden	2.1

3.2.1.3 Vorbereitung

Tabelle 7: Arbeitspakete Vorbereitung

ID	Name	Definition of Done	Aufwand	Abhängigkeit
3.1	Vorkenntnisse	Die persönlichen Vorkenntnisse wurden evaluiert und dienen als Grundlage für den finalen Technologie-Stack.	2 Stunden	Planung beendet
3.2	Definition der Systemarchitektur	Die architektonischen Leitplanken sind klar dokumentiert und ausformuliert mit einer Entscheidungsmatrix.	8 Stunden	Planung beendet
3.3	Technologie-Evaluation	Der finale Technologie-Stack ist geprüft und festgelegt	6 Stunden	3.1, 3.2
3.4	Cyber Security Konzepte ausarbeiten	Die Sicherheitsarchitektur ist ausgearbeitet auf der Basis des Technologie-Stack. Schutzmassnahmen gegen OWASP-Schwachstellen sind definiert.	6 Stunden	3.3
3.5	Datenschutz und rechtliche Aspekte	Sicherstellen, dass der Datenschutz korrekt berücksichtigt wird. Es besteht ein Unterkapitel, welches sich mit dem DSG befasst und der Umgang mit den Nutzerdaten ist ausformuliert.	4 Stunden	3.3
3.6	Teststrategie	Die Teststrategie für Backend und Frontend ist konzeptionell dokumentiert.	4 Stunden	3.3
3.7	Versions-verwaltung	Der Einsatz von Git ist klar definiert und die Branching-Strategie verbindlich festgehalten.	2 Stunden	3.3

3.8	CI/CD-Pipeline	Die potenziellen Werkzeuge für die spätere CI/CD-Pipeline sind definiert und der grobe Ablauf ist festgehalten.	4 Stunden	3.7
3.9	UI/UX-Strategie	Eine geeignete Komponentenbibliothek ist ausgewählt, um den Custom-Styling-Aufwand im Frontend zu minimieren.	4 Stunden	3.3

3.2.1.4 Anforderungsanalyse

Tabelle 8: Arbeitspakete Anforderungsanalyse

ID	Name	Definition of Done	Aufwand	Abhängigkeit
4.1	Geschäftsanwendungsfälle definieren	Die wichtigsten Geschäftsprozesse sind aus der fachlichen Perspektive beschrieben.	4 Stunden	Vorbereitung beendet
4.2	Systemanwendungsfälle ableiten	Ein vollständiges Use-Case-Diagramm ist erstellt. Systemanwendungsfälle im Detail dokumentiert.	8 Stunden	4.1
4.3	Funktionale Anforderungen erarbeiten	Aus den Systemanwendungsfällen ist eine vollständige Liste der funktionalen Anforderungen erstellt.	8 Stunden	4.2
4.4	Nicht-funktionale Anforderungen definieren	Die qualitativen Anforderungen wie Performance, Security, Usability etc. sind dokumentiert.	8 Stunden	4.2
4.5	Abnahme Anforderungskatalog	Es wurde ein Review des gesamten Anforderungskatalogs durchgeführt. Die Vollständigkeit und Testbarkeit haben keine Widersprüche. Es ist eine eingefrorene Baseline für die Konzeptionsphase vorhanden.	2 Stunden	4.3, 4.4

3.2.1.5 Konzeption

Tabelle 9: Arbeitspakete Konzeption

ID	Name	Definition of Done	Aufwand	Abhängigkeit
5.1	Datenbankmodellierung (ERM & Relational)	Das konzeptionelle ER-Diagramm sowie das physische relationale Datenbankmodell (inklusive Tabellen, Datentypen, Constraints und Beziehungen) sind finalisiert und dokumentiert.	6 Stunden	Anforderungsanalyse beendet
5.2	Klassen-diagramme	Die Backend-Architektur ist in Klassendiagrammen modelliert. Es ist festgehalten welche Controller, Services, Repositories, Entities und DTOs es gibt.	8 Stunden	5.1

		Ebenfalls werden die wichtigsten Business-Regeln festgehalten.		
5.3	REST-API Design & Schnittstellen-vertrag	Alle benötigten Endpoints, HTTP-Methoden, HTTP-Statuscodes sowie die Request- und Response-Payloads (DTOs) sind vollständig dokumentiert. Ursprünglich war die Swagger Spezifikation auch schon geplant. Dies wird aber erst bei der Entwicklung umgesetzt. (6.4)	6 Stunden	5.2
5.4	Modellierung der Abläufe	Für komplexere Use Cases wie die Live-Protokollierung oder die Trainingsplanung sind Aktivitätsdiagramme erstellt. Für technische Abläufe wie z.B. den Login-Prozess wurden Sequenzdiagramme erstellt. Ein Zustandsdiagramm für den Flow des Status ist erstellt.	10 Stunden	5.3
5.5	GUI-Prototyping & Navigationskonzept	Navigationskonzept, Screens mit ihren wichtigsten Elementen und der Screen-Flow sind dokumentiert. Auf Mockups wird verzichtet, weil die Standardkomponenten die Gestaltung vorgeben.	8 Stunden	5.4
5.6	Review Technischer Blueprint	Prüfen aller Diagramme, Schnittstellen und Flowcharts auf eine durchgehende Konsistenz. Es müssen alle Anforderungen abgedeckt sein.	4 Stunden	5.1-5.5

3.2.1.6 Entwicklung

Tabelle 10: Arbeitspakete Entwicklung

ID	Name	Definition of Done	Aufwand	Abhängigkeit
6.1	Setup & Infrastruktur (Lokal)	Die Entwicklungsumgebung, das Git-Repository und die lokalen Docker-Container sind initialisiert und lauffähig.	6 Stunden	Konzeption beendet
6.2	Vertical Slice (Proof of Concept)	Ein minimaler End-to-End Anwendungsfall funktioniert vom Frontend über die REST-API bis zur Datenbank.	12 Stunden	6.1
6.3	User Management & Security	Registrierung, Login, JWT-Authentifizierung und das Löschkonzept sind im Backend gemäss den Vorgaben vollständig	16 Stunden	6.2

		implementiert und mit Unit-Tests abgesichert.		
6.4	Backend fertigstellen	Die komplette Geschäftslogik für den Übungskatalog, Trainingsroutinen etc. ist implementiert. Die REST-Endpunkte stehen wie in der Konzeption festgelegt zur Verfügung für das Frontend. Die OpenAPI-Spezifikation ist mit springdoc erzeugt.	16 Stunden	6.3
6.5	Frontend Setup & UI-Bibliothek	Das React-Projekt ist mit dem gewählten Framework und der Komponentenbibliothek konfiguriert. Das State-Management und Routing-Grundgerüst stehen.	4 Stunden	6.2
6.6	Frontend Authentifizierung	Die UI für Login, Registrierung und Profilverwaltung (inkl. Datenlöschung) ist umgesetzt und konsumiert die REST-API erfolgreich.	12 Stunden	6.3, 6.5
6.7	Frontend fertigstellen	Die UI-Komponenten für das Erstellen/Bearbeiten von Routinen, den Übungskatalog und die Live-Protokollierung sind vollständig angebunden und nutzbar.	20 Stunden	6.4, 6.6
6.8	Testing und Bugfixing	Die gesamte Applikation ist getestet und identifizierte Bugs sind behoben. Alle Muss-Ziele sind erfüllt.	8 Stunden	6.7

3.2.1.7 Deployment

Tabelle 11: Arbeitspakete Deployment

ID	Name	Definition of Done	Aufwand	Abhängigkeit
7.1	Server einrichten	Der Server ist konfiguriert und Docker ist installiert.	8 Stunden	Entwicklung beendet
7.2	Dockerisierung für Produktion	Die produktiven Dockerfiles sind geschrieben. Die ganze Applikation lauft fehlerfrei mit Docker Compose.	8 Stunden	7.1
7.3	Aufbau der CI/CD-Pipeline	Eine automatisierte Pipeline ist eingerichtet. Bei einem Push in den Main-Branch wird der Code getestet, gebaut und automatisch auf den Server deployed.	8 Stunden	7.2
7.4	Domain & SSL-Zertifikat (HTTPS)	Die Applikation ist über eine feste Domain oder Subdomain erreichbar. Die gesamte	4 Stunden	7.3

		Kommunikation ist zwingend über HTTPS verschlüsselt.		
7.5	Go-Live & Finaler Smoke-Test	Die Applikation ist öffentlich erreichbar. Ein abschliessender End-to-End-Test in der Produktionsumgebung war erfolgreich. Meilenstein 3 ist erreicht.	4 Stunden	7.4

3.2.1.8 Abschluss

Tabelle 12: Arbeitspakete Abschluss

ID	Name	Definition of Done	Aufwand	Abhängigkeit
8.1	Code-Bereinigung	Der Code (Backend & Frontend) ist bereinigt.	4 Stunden	Deployment beendet
8.2	Erstellung der Benutzer- und Betriebsdokumentation	Eine kurze Anleitung für die Endnutzer sowie eine technische Betriebsdokumentation (für das Starten/Stoppen der Docker-Container auf dem Server) ist verfasst.	6 Stunden	8.1
8.3	Projektreflexion	Es gibt eine Reflexion die eine Erfolgskontrolle und eine persönliche Reflexion beinhaltet.	8 Stunden	8.2
8.4	Schlusswort und Management Summary	Das Schlusswort ist verfasst. Das Management Summary am Anfang des Dokuments ist mit den finalen Ergebnissen und Erkenntnissen aktualisiert.	4 Stunden	8.3
8.5	Formale Prüfung und Formatierung	Das gesamte Word-Dokument entspricht den Vorgaben der TEKO. Alle Verzeichnisse, Anhänge, Quellenangaben etc. sind vollständig. Die Rechtschreibung wurde geprüft.	4 Stunden	8.4

3.2.1.9 Übersicht der Arbeitspakete

Tabelle 13: Aufwand pro Phase

Phase	Gesamtaufwand
Initialisierung	24 Stunden
Planung	14 Stunden
Vorbereitung	40 Stunden
Anforderungsanalyse	30 Stunden
Konzeption	42 Stunden
Entwicklung	94 Stunden
Deployment	32 Stunden
Abschluss	26 Stunden
Total	302 Stunden

4 Vorbereitung/Analyse

4.1 Vorkenntnisse

Als Basis für die spätere Technologie-Evaluation macht es Sinn, kurz einen Blick auf den aktuellen Wissensstand zu werfen.

Backend & Datenbanken

Tabelle 14: Vorkenntnisse Backend und Datenbanken

Technologie	Beschreibung	Bisherige Erfahrung
Java	Objektorientierte Programmiersprache, die primär für die Geschäftslogik im Backend eingesetzt wird.	Sehr hohe Erfahrung
SQL	Datenbanksprache zur Erstellung, Abfrage und Manipulation von relationalen Datenbanken.	Sehr hohe Erfahrung
PostgreSQL	Relationales Datenbankmanagementsystem.	Sehr hohe Erfahrung
Maven	Build-Management-Tool für Java. Verwaltung von Dependencies.	Gute Erfahrung
Spring Boot	Java-basiertes Framework für Standalone-Applikationen und REST-APIs	Gute Erfahrung
Testing-Frameworks	JUnit oder Mockito für Code-Qualität	Gute Erfahrung
Schnittstellendesign	RESTful API Design und JSON	Sehr hohe Erfahrung
Web-Security Grundlagen	Authentifizierungskonzepte JWT	Grundkenntnisse
WildFly Application Server	Applikationsserver für Java Enterprise und Jakarta EE Applikationen. Früher bekannt als JBoss.	Grundkenntnisse
GlassFish	Kostenloser Open-Source-Applikationsserver für Java Enterprise und Jakarta EE Applikationen. Kommt von der Eclipse Foundation	Wenig Grundkenntnisse
MongoDB	Dokumentenorientiertes NoSQL Datenbankmanagementsystem. Vor allem für Sammlungen von JSON-ähnlichen Dokumenten gut.	Keine Erfahrung
C#	Objektorientierte Programmiersprache für .NET-Plattformen und Datenbankanwendungen geeignet.	Keine Erfahrung

Frontend & UI

Tabelle 15: Vorkenntnisse Frontend

Technologie	Beschreibung	Bisherige Erfahrung
React	JavaScript-Bibliothek für die Erstellung von modernen Benutzeroberflächen.	Grundkenntnisse
UI-Komponentenbibliotheken	Vorgefertigte responsive UI-Komponenten für eine schnelle Frontend-Entwicklung	Geringe Erfahrung
HTML/CSS	Optische Gestaltung von Webseiten.	Gute Erfahrung
Testing-Frameworks	Testing-Frameworks für Frontend	Grundkenntnisse
JavaScript	Programmiersprache, um Webseiten zu steuern und zu verändern.	Grundkenntnisse
TypeScript	TypeScript fügt eine statische Typisierung zu JavaScript hinzu.	Grundkenntnisse
Angular	JavaScript-basiertes Framework für die Entwicklung von Web-Anwendungen.	Keine Erfahrung
Figma	Software zur Gestaltung von Benutzeroberflächen und Prototypen.	Geringe Erfahrung

Infrastruktur, Deployment & Tools

Tabelle 16: Vorkenntnisse Infrastruktur und Tools

Technologie	Beschreibung	Bisherige Erfahrung
Docker	Containerisierung von Applikationen.	Geringe Erfahrung
CI/CD-Tools	Werkzeuge zur Automatisierung von Deployment.	Geringe Erfahrung
Versionsverwaltung Git	Versionsverwaltungssystem für die Code-Verwaltung.	Sehr hohe Erfahrung
Server Administration	Umgang mit Servern für das Hosting der Applikation	Grundkenntnisse
Reverse Proxy	Webserver zur Weiterleitung von Anfragen an die Docker Container	Sehr geringe Erfahrung

4.2 Definition der Systemarchitektur

Mit den Vorkenntnissen aus dem vorherigen Kapitel hat man eine gute Vorlage, auf der man aufbauen kann. Bevor man aber die einzelnen Technologien wählen kann braucht es den strukturellen Rahmen. Laut Definition of Done dieses Kapitels sollen die architektonischen Leitplanken dokumentiert sein und eine Entscheidungsmatrix erstellt werden. Dies wird

gemacht, bevor man anfängt zu entwickeln. Man erarbeitet zuerst einen grundsätzlicher Aufbau des Systems, um das Zusammenspiel der Teile und Art der Datenhaltung zu eruieren. Die Entscheidungen in diesem Kapitel gehen mehr in die Richtung von «relationale Datenbank vs. dokumentenorientiert» und nicht schon «PostgreSQL vs. MongoDB». Die effektive Technologie-Evaluation ist im nächsten Kapitel angedacht.

Die Entscheidungen werden nicht frei gewählt, sondern aus Architekturtreibern abgeleitet. Diese stammen aus den bereits erarbeiteten Muss/Kann/Nicht-Zielen, Projektvision, Risikoanalyse, Stakeholder Analyse und Wirtschaftlichkeitsanalyse.

Sobald man die Architekturtreiber sauber strukturiert hat, wird der Systemkontext festgelegt. Also was sind die Systemgrenzen, Nutzer und externe Dienste dieser Applikation.

Es gibt Architekturentscheidungen zu treffen für den Backend-Stil, Frontend-Ansatz, Deployment-Topologie und Persistenzlösung. Diese werden mit einer Bewertungsmatrix jeweils gut begründet.

Auch Querschnittsthemen wie Fehlerbehandlung, Logging und Konfigurationen werden behandelt. Es gibt aber keine Detailentwürfe, das ist ganz wichtig! Diese folgen erst in der Konzeption. Das Ergebnis dieses Kapitels sind verbindliche Leitplanken der Architektur für die spätere Konzeption.

4.2.1 Architekturtreiber

Tabelle 17: Architekturtreiber

ID	Architekturtreiber	Herkunft	Prio
T1	Datenhoheit inkl. vollständiger Löschbarkeit der Nutzerdaten	Muss-Ziel Löschkonzept, Vision	Hoch
T2	Eine Codebasis für Smartphone und Desktop, responsive	Muss-Ziel, Nicht-Ziel «keine native App»	Hoch
T3	Betrieb durch eine Einzelperson bei ca. 10–15 CHF/Monat	Betriebskosten, Stakeholder Provider	Hoch
T4	Umsetzbarkeit im Zeitbudget einer Person bis 28.10.2026	Risiko Einzelentwickler, Risiko Over-Engineering	Hoch
T5	Nutzung vorhandener Kernkompetenzen (Java, SQL, PostgreSQL)	Vorkenntnisse, Risiko Frontend-Aufwand	Hoch
T6	Erweiterbarkeit für Kann-Ziele mit vertretbarem Aufwand	Kann-Ziele, Vision	Mittel
T7	Serverseitige Authentifizierung und Autorisierung	Muss-Ziel Login, Cyber Security Aspekte	Hoch
T8	Testbarkeit von Backend und Frontend	Teststrategie, Risiko Bias beim Testing	Mittel
T9	Flüssige Bedienung während der Live-Protokollierung	Muss-Ziel Live-Protokollierung, Projektvision	Hoch

4.2.2 Systemkontext

Der Systemkontext gibt an, was selbst entwickelt wird und was nicht.

Als Nutzer hat man aktuell nur die Rolle Sportler, der Trainer ist ein Kann-Ziel und muss nicht explizit erwähnt werden.

Externe Dienste: Hosting-Provider, Domain/TLS, möglicher Mailversand, möglicher Objektspeicher für den Medien-Upload.

Im Diagramm sind die effektiv wichtigen Teile des Systems mit grün markiert. Was jeweils aus den «Kann-Zielen» kommt, ist orange.

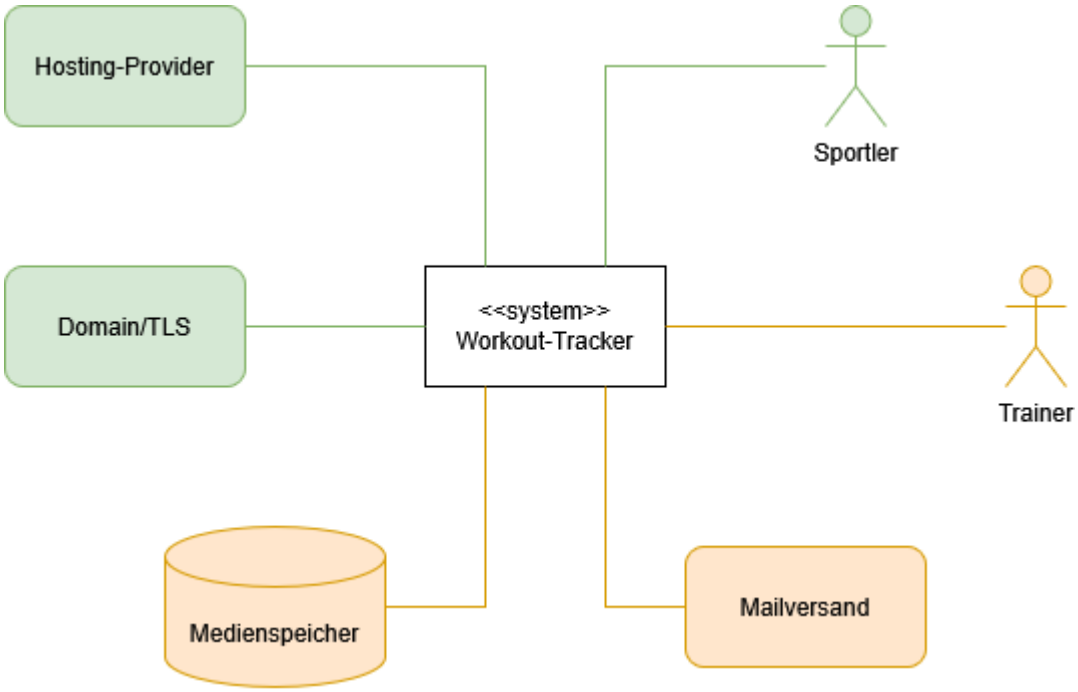


Abbildung 4: Systemkontext

4.2.3 Entscheidungen

Es werden für die wichtigsten Entscheidungen jeweils die möglichen Kandidaten aufgegliedert. Anschliessend werden diese anhand einer Entscheidungsmatrix evaluiert. Die relevanten definierten Architekturtreiber werden jeweils als Kriterium der Bewertung genommen.

Die Berechnung erfolgt anhand des Gewichts der Priorität (Hoch = 2, Mittel = 1) und einer Bewertung von 1, 3 oder 6 (1 = Gering, 3 = Mittel, 6 = Hoch) pro Kandidaten. Die 6 bedeutet also, dass der Kandidat den jeweiligen Architekturtreiber sehr gut erfüllt. Es wird schlussendlich jeweils das Produkt pro Gewichtung und Bewertung berechnet. Anschliessend summiert man pro Treiber diese Werte. Somit hat man pro Kandidaten einen Gesamtwert.

Die Reihenfolge der Entscheidungen ist nicht einfach beliebig gewählt worden. Sie folgt dem Aufbau von unten nach oben. Die Struktur der Daten gibt einen grossen Teil des Backends vor. Das Backend bestimmt dann, wie das Frontend angebunden werden kann. Erst aus allen drei Teilen ergibt sich schlussendlich, wie das System betrieben wird.

4.2.3.1 E1 - Persistenzansatz

Wie sollen die Trainingsdaten strukturiert und gespeichert werden?

Kandidaten:

- Relational: Benutzer, Übung, Routine, Trainingseinheit und Satz sind alles eigene Entitäten mit Fremdschlüsseln zueinander. Das Schema ist klar definiert. Auswertungen erfolgen über Joins und Aggregatsfunktionen.
- Dokumentenorientiert: Eine Trainingseinheit wird inklusive sämtlicher Sätze, Wiederholungen etc. in einem einzigen Dokument abgespeichert. Es gibt kein erzwungenes Schema. Die Verweise der Dokumente werden in der Applikation aufgelöst.

Tabelle 18: Entscheidungsmatrix Persistenzansatz

Kriterium	Gewicht	Relational	Dokumentenorientiert
T1 (Datenhoheit inkl. Löschbarkeit)	2	6 (12)	1 (2)
T4 (Zeitbudget)	2	6 (12)	1 (2)
T5 (Nutzung der vorhandenen Kernkompetenzen)	2	6 (12)	1 (2)
T6 (Erweiterbarkeit)	1	3 (3)	6 (6)
T8 (Testbarkeit)	1	6 (6)	3 (3)
Total		45	15

Nicht bewertete Kriterien:

- T2 eine Codebasis: Reine Frontend-Frage.
- T3 Betrieb durch eine Person: Beide Kandidaten werden als eigenständiger Service betrieben.
- T7 Serverseitige Authentifizierung und Autorisierung: Wird im Backend umgesetzt und ist hier nicht zu berücksichtigen.
- T9 flüssige Bedienung während des Trainings: Reine Frontend-Frage.

Die Entscheidung anhand der Matrix fällt für dieses Projekt einseitig aus. Trotzdem ist es wichtig, die Stärken des dokumentenorientierten Modells anzuerkennen. Man könnte jede Trainingseinheit als komplettes Dokument zusammengefasst ablegen. Somit entfällt das Zusammensetzen über mehrere Tabellen. Neue Angaben an einer Übung sind auch ohne Schemaänderung möglich.

Diese Argumente reichen aber nicht aus. Es sind verschiedene Abfragen vorgesehen, welche quer über die Trainingseinheiten gehen, um diese zu vergleichen. Mit der dokumentenorientierten Datenhaltung sind viele Werte entweder doppelt vorhanden oder müssen kompliziert in der Applikation zusammengeführt werden. Auch die vollständige Löschbarkeit ist nicht besonders praktisch. Wenn man vergisst, eine Stelle abzudecken, lässt man unbemerkt Daten bestehen, welche eigentlich gelöscht werden müssen. Der direkte Bezug zur Datenhoheit ist somit nicht gegeben.

Für diese Diplomarbeit wird eine **relationale Datenbank** gewählt. Mit einem relationalen Datenmodell können die Normalisierungsformen angewendet werden. Jede fachliche Entität hat ihre eigene Tabelle mit einem eindeutigen Schlüssel. Die Referenzen sind zwingend in der Datenbank über Fremdschlüssel zu führen. Mit Fremdschlüsselketten können alle Daten von

einem Benutzer identifiziert und sauber gelöscht werden. Das Schema bietet eine eindeutige reproduzierbare Datenbank.

4.2.3.2 E2 – Backend-Stil

Wie wird die Geschäftslogik im Backend strukturiert und ausgeliefert? Dabei werden Schichtenmonolith, Modularer Monolith und Microservices verglichen [6].

Kandidaten:

- Schichtenmonolith: Aufteilung der Pakete in horizontale Schichten (Controller, Business Logik, Persistenz). Es ist eine hohe Kopplung vorhanden. Die gesamte Anwendung wird als eine einzige Einheit gebaut und bereitgestellt.
- Modularer Monolith: Vertikale Aufteilung nach fachlichen Modulen. Benutzerverwaltung, Übungskatalog, Routinen, Trainingsprotokoll etc. haben jeweils eine definierte Schnittstelle nach aussen und die internen Klassen bleiben gekapselt. Wird als einzige Einheit deployed.
- Microservices: Ähnlich wie der modulare Monolith, aber jedes fachliche Modul ist ein eigenständiges Deployment mit eigener Datenhaltung.

Matrix:

Tabelle 19: Entscheidungsmatrix Backend-Stil

Kriterium	Gewicht	Schichtenmonolith	Modularer Monolith	Microservices
T3 (Betrieb durch eine Person)	2	6 (12)	6 (12)	1 (2)
T4 (Zeitbudget)	2	6 (12)	3 (6)	1 (2)
T5 (Nutzung der vorhandener Kernkompetenzen)	2	6 (12)	6 (12)	3 (6)
T6 (Erweiterbarkeit)	1	3 (3)	6 (6)	6 (6)
T8 (Testbarkeit)	1	3 (3)	6 (6)	3 (3)
Total		42	42	19

Nicht bewertete Kriterien:

- T1: Datenhoheit: unabhängig vom Architekturstil, die Löschbarkeit hängt am Datenmodell und Persistenz.
- T2 eine Codebasis: Reine Frontend-Frage.
- T7 Serverseitige Authentifizierung und Autorisierung: Bei allen drei Kandidaten im Backend und somit gleichwertig.
- T9 flüssige Bedienung während des Trainings: Reine Frontend-Frage.

Der Schichtenmonolith und Modularer Monolith haben mit dem gleichen Ergebnis abgeschnitten. Ein Microservice kommt jedoch nicht in Frage, wenn es nach der Entscheidungsmatrix geht. Der Aufwand von einem Microservice entsteht nicht unbedingt in der Fachlichkeit, sondern in der Infrastruktur und Netzwerkkommunikation. Bei einem Projekt mit einem einzigen Entwickler und einem knappen Zeitfenster ist das schlichtweg kein gangbarer Weg. Wenn man ein grösseres Team hätte, wäre die Bewertung ganz anders.

Der Schichtenmonolith spart an Aufwand in der Startphase, weil man nicht an den fachlichen vertikalen Schnitt denken muss. Erst wenn viele Erweiterungen dazukommen, hat man einen Nachteil mit der Erweiterbarkeit und Testbarkeit.

Wenn man aber mit einem Schichtenmonolithen anfängt, wird es schwierig später daraus einen modularen Monolith zu erstellen, da es gewachsene Abhängigkeiten gibt auf verschiedensten Codezeilen und Ebenen.

Da die Matrix zwischen den beiden Monolithen keinen Unterschied ergibt, braucht es ein weiteres Kriterium für den Entscheid. Das Entscheidende Tie-Break-Kriterium hier wurde in der Zieldefinition schon klar erwähnt. Die Muss-Ziele haben Priorität. Die Kann-Ziele werden gemäss Planung ebenfalls erst bei verbleibender Zeit angegangen. Somit ist ein Schichtenmonolith die beste Wahl.

Für diese Diplomarbeit wird der **Schichtenmonolith** gewählt. Mit diesem Entscheid ist ausschliesslich der Architekturstil festgelegt. Welches Framework dafür verwendet wird, ist erst Teil der Technologie-Evaluation.

Die horizontalen Schichten von Controller/Business Logik und Persistenz sind zwingend einzuhalten. Innerhalb jeder Schicht gibt es eigene Pakete je nach fachlichem Bereich (Benutzerverwaltung, Übungskatalog, Trainingsroutinen, Trainingsprotokoll). Zugriffe zwischen den fachlichen Bereichen sind zulässig, sofern sie über die Service-Schicht des jeweiligen Bereichs erfolgen. Ein direkter Zugriff auf Repositories oder Entitäten eines anderen Bereichs wird vermieden. Somit sind die fachlichen Grenzen vorhanden, auch wenn sie technisch nicht erzwungen werden. Wenn man also zu einem späteren Zeitpunkt eine Migration zu einem Modularen Monolith in Erwägung zieht, hat man eine gewisse Grundlage. Die fachliche Paketierung erlaub es auch, das Testkonzept anhand dieser Bereiche zu organisieren.

Bewusst in Kauf genommene Nachteile:

- Keine Skalierung einzelner Bereiche wie es mit Microservices möglich wäre
- Modulgrenzen sind nicht technisch erzwungen
- Mit wachsendem Umfang steigt das Risiko von unerwünschten Abhängigkeiten.

4.2.3.3 E3 – Frontend-Ansatz

Wie wird die Benutzeroberfläche erzeugt und ausgeliefert? Es wird der Unterschied zwischen serverseitigem Rendering, einem hybriden Ansatz mit partiellen Updates und einer Single-Page-Application (SPA) analysiert.

Kandidaten:

- Serverseitiges Rendering: Das Backend erzeugt die HTML-Templates je nach Aufruf. Jede Interaktion des Users führt zu einem Seitenwechsel.
- Hybrider Ansatz: Das Backend erzeugt weiterhin das HTML, einzelne Bereiche der Seite werden aber mit JavaScript-Bausteinen nachgeladen und ausgetauscht.
- Single-Page-Application (SPA): Das Frontend ist ein eigenständiges Projekt und Deployment. Das Backend liefert ausschliesslich Daten über eine REST-Schnittstelle

Matrix:

Tabelle 20: Entscheidungsmatrix Frontend-Ansatz

Kriterium	Gewicht	Serverseitiges Rendering	Hybrider Ansatz	SPA
T5 (Nutzung der vorhandenen Kernkompetenzen)	2	3 (6)	3 (6)	3 (6)
T6 (Erweiterbarkeit)	1	1 (1)	3 (3)	6 (6)
T8 (Testbarkeit)	1	3 (3)	3 (3)	6 (6)
T9 (Bedienung während des Trainings)	2	1 (2)	3 (6)	6 (12)
Total		12	18	30

Nicht bewertete Kriterien:

- T2 eine Codebasis: Für alle Kandidaten gleichwertig.
- T4 Zeitbudget: Der Mehraufwand für die Entwicklung wurde im Arbeitspaket dazu schon ausgiebig eingeplant. Somit ist es für diese Entscheidung kein Kriterium.
- T7 Serverseitige Authentifizierung und Autorisierung: Bei allen drei Kandidaten im Backend und somit gleichwertig.

Das Serverseitige Rendering ist vielleicht in der Startphase der schnellste Weg, aber sobald man beim Protokollieren einer Trainingseinheit in kurzer Folge mehrere Abfragen auslöst, bei denen die Seite neu geladen wird, hat man ein Problem, welches nicht vertretbar ist. Ebenfalls sind Kann-Ziele wie Verlaufscharts oder grafische Dashboards auch schwierig umzusetzen mit dieser Architektur.

Der hybride Ansatz löst einige Probleme des Serverseitigen Renderings. Die Bedingungen während dem Training sind schon spürbar besser. Für diesen Ansatz sind aber auch keine wirklichen Vorkenntnisse vorhanden.

Eine Single-Page-Application, welche über eine REST-Schnittstelle auf das Backend zugreift, ist der absolute Industriestandard. Ebenfalls sind schon Grundkenntnisse in React und TypeScript vorhanden. Dieser Ansatz verursacht aber auch etwas mehr an Aufwand, weil eine zweite Codebasis mit eigener Build-Kette und Deployment aufgesetzt werden muss. Die getrennte Codebasis erlaubt es aber wiederum das Frontend über Komponententests isoliert zu prüfen, während das Backend über die Schnittstelle getestet wird.

Für diese Diplomarbeit wird die **Single-Page-Application** gewählt. Welches Framework und welche Komponentenbibliothek eingesetzt werden, ist Teil der Technologie-Evaluation.

Das Frontend enthält keine Geschäftslogik. Die Eingaben dürfen im Frontend validiert werden für eine freundliche Benutzerführung. Die verbindliche Validierung und sämtliche Autorisierungsentscheide erfolgen jedoch immer serverseitig. Es muss eine typisierte Sprache benutzt werden, damit der Schnittstellenvertrag zum Backend überprüfbar bleibt. Mit einem Mobile-First-Layout wird die Benutzung vom Smartphone bevorzugt. Im Desktop greifen dann die responsiven Komponenten.

Bewusst in Kauf genommene Nachteile:

- Zwei Codebasen mit getrennten Build-Ketten welche zu mehr Aufwand führen.

- Änderungen an der Schnittstelle betreffen immer beide Seiten und erfordern Disziplin bei der Versionierung.

4.2.3.4 E4 - Deployment-Topologie

Wo und in welcher Struktur werden die Bestandteile der Applikation betrieben?

Kandidaten:

- Verwaltete Plattform: Die Bestandteile werden einem Plattformdienst übergeben. Der Anbieter verwaltet die Laufzeitumgebung, Zertifikate, Netzwerk. Man hat kein eigenes Betriebssystem und keine Serveradministration.
- Einzelner Server: Das Frontend, Backend und Datenbank als Container auf einem Host. Ein Reverse Proxy nimmt die Anfragen entgegen. Das Betriebssystem und die Laufzeitumgebung sind in eigener Verantwortung.
- Container-Orchestrierung: Die verschiedenen Bestandteile als Arbeitslasten auf einem Cluster deployen. Das Cluster übernimmt die Skalierung der einzelnen Nodes.

Matrix:

Tabelle 21: Entscheidungsmatrix Deployment-Topologie

Kriterium	Gewicht	Verwaltete Plattform	Einzelner Server	Container-Orchestrierung
T1 (Datenhoheit inkl. Löschbarkeit)	2	3 (6)	6 (12)	6 (12)
T3 (Betrieb durch eine Person, 10–15 CHF/Monat)	2	3 (6)	6 (12)	1 (2)
T4 (Zeitbudget)	2	6 (12)	3 (6)	1 (2)
T6 (Erweiterbarkeit)	1	3 (3)	6 (6)	6 (6)
Total		27	36	22

Nicht bewertete Kriterien:

- T2 eine Codebasis: von der Betriebsform unabhängig.
- T5 Kernkompetenzen: bezieht sich auf Java/SQL/PostgreSQL/React, nicht auf Betrieb und Infrastruktur.
- T7 Serverseitige Authentifizierung und Autorisierung: Bei allen drei Kandidaten im Backend und somit gleichwertig.
- T8 Testbarkeit: wird von Backend-Stil und Frontend-Ansatz bestimmt.
- T9 flüssige Bedienung: Frontend-Thema.

Eine verwaltete Plattform ist am schnellsten in der Aufsetzung. Man hat keinen Server, den man betreiben muss und senkt somit das Risiko einer Deployment-Verzögerung. Dafür sind solche Lösungen meist die teuerste Variante. Mit separatem Deployment für Frontend, Backend und Datenbank würde das den Rahmen sprengen. Die Daten und die Sicherung sind komplett im Verantwortungsbereich des Anbieters, was keine passende Lösung ist.

Mit einer Container-Orchestrierung hätte man die höchste Erweiterbarkeit und eine technisch saubere Struktur. Dieser Ansatz wäre sehr spannend zu verfolgen, wenn man auch einen Microservice-Ansatz im Backend verfolgen würde. Die Kosten sind zwar etwas tiefer als bei einer verwalteten Plattform, kommen aber nicht an einen einzelnen Server ran. Ebenfalls ist keine nützliche Erfahrung in der Orchestrierung vorhanden. Nur schon zur Containerisierung ist geringe Erfahrung vorhanden. Die Applikation wird im Rahmen dieser Diplomarbeit von einer

Person genutzt. Man nimmt für die Performance keine Rücksicht darauf, dass in Zukunft auch weitere Sportler die Plattform verwenden werden. Die Ausfallsicherheit oder eine horizontale Skalierung bringen keinen erkennbaren Nutzen. Es wäre auch ein gutes Beispiel für das Over-Engineering, welches in der Risikoanalyse erwähnt wurde.

Für diese Diplomarbeit wird ein **einzelner Server** aufgesetzt. Somit hat man einen Host und die Kosten sind innerhalb des Rahmens. Die Datenablage und Sicherungen sind in den eigenen Händen. Der Mehraufwand entsteht bei der Serverkonfiguration und Reverse Proxy, wo schwache Vorkenntnisse vorhanden sind. Ein einmaliger Aufwand ist aber im Arbeitspaket des Deployments schon eingeplant und somit keine grosse Gefahr.

Es müssen alle Bestandteile als Container auf demselben Host liegen. Der Zugriff ist ausschliesslich über den Reverse Proxy, TLS wird dort terminiert. Das Backend und die Datenbank sind somit von aussen nicht direkt erreichbar. Die Zugangsdaten und Schlüssel werden zur Laufzeit bereitgestellt. Das Deployment ist reproduzierbar, also ein Neuaufbau aus dem Repository auf einem leeren Host wäre möglich. Ein automatischer Backup-Prozess wird im Rahmen dieser Arbeit nicht berücksichtigt. Backups erfolgen höchstens manuell.

Bewusst in Kauf genommene Nachteile:

- Ausfall des Hosts bedeutet Ausfall der Applikation
- Keine horizontale Skalierung
- Ein Konfigurationsfehler auf dem Host betrifft direkt auch das ganze System.

4.2.4 Bausteinsicht

Die Entscheidungen E1-E4 werden in einer Bausteineinsicht visuell dargestellt. Hierbei handelt es sich um eine hohe Ebene, bei denen man nicht auf die einzelnen Klassen eingeht. Das passiert erst in der Konzeption

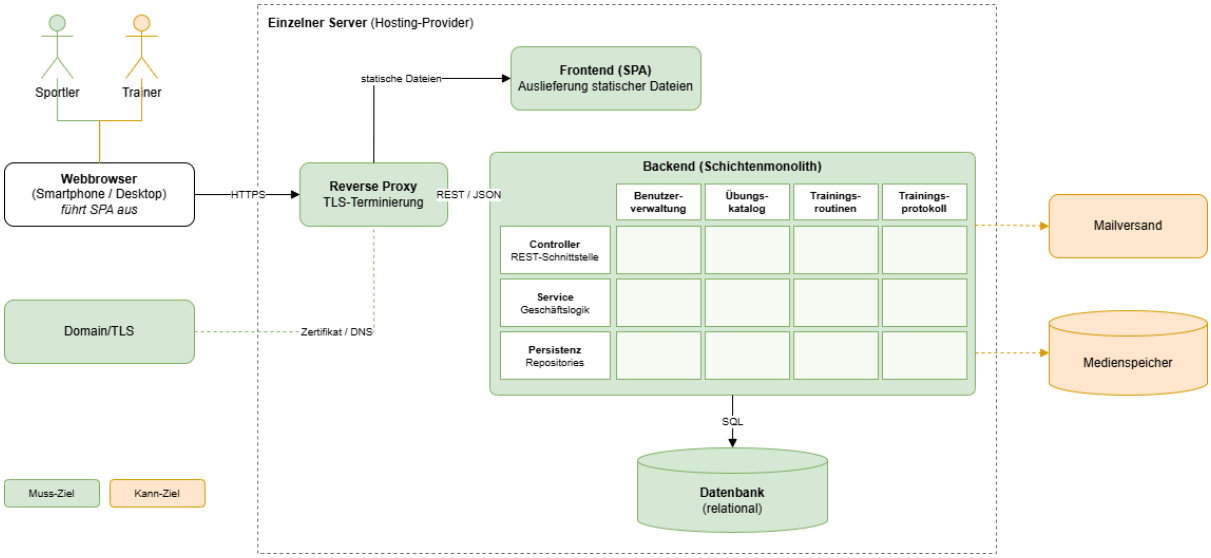


Abbildung 5: Bausteinsicht

Herleitungen aus den Entscheidungen:

- E4: Alles läuft auf einem Host, der einzige Zugang ist der Reverse Proxy.
- E3: Der Frontend-Container liefert nur die statischen Dateien aus. Die SPA selbst läuft im Browser.

- E2: Das Backend ist in drei Schichten (Controller, Service, Persistenz) und vier fachliche Bereiche gegliedert, darum das Raster.
- E1: Eine relationale Datenbank, die nur das Backend erreicht.

Kann-Ziele: Mailversand, Medienspeicher und Trainer sind orange und gestrichelt. Sie werden schon aufgezeigt, werden aber nicht zwingend umgesetzt.

4.2.5 Querschnittsthemen als Leitplanken

Dies sind Themen, welche über die ganze Applikation laufen und keinem spezifischen Baustein zugeordnet werden können. Daher werden sie einheitlich festgelegt. Security/Authentifizierung, Datenschutz und Tests werden bewusst noch nicht behandelt, da sie erst Teil der nächsten Kapitel sind.

4.2.5.1 Fehlerbehandlung

Die Fehler werden zentral im Backend behandelt und nicht pro Bereich oder Schicht einzeln. Mit einem einheitlichen Fehlerformat wird ein Standard festgelegt. Dieser besteht aus einem Statuscode, einer verständlichen Meldung und wenn nötig noch weitere Details. Die Stacktraces und interne Details bleiben im Backend und gehen nicht an den Client. Der Nutzer sieht im Frontend also nur eine verständliche Meldung mit einem leserlichen Fehlertext.

4.2.5.2 Logging

Das Logging ist wie die Fehlerbehandlung auch nur im Backend, und zwar mit festen Log-Levels. Im Log stehen keine sensiblen Daten wie Passwörter, Tokens oder Trainingsinhalte. Höchstens IDs zu den jeweiligen Datensätzen. Die Logs werden nur begrenzt aufbewahrt. Jeweils 14 Tage.

4.2.5.3 API-Stil

Die API-Calls werden mit JSON abgehandelt. Die HTTP-Methoden entsprechen ihrer Bedeutung und es werden die korrekten Statuscodes verwendet. Alle Endpunkte liegen unter einem gemeinsamen Präfix (/api). Der Schnittstellenvertrag wird dokumentiert und ist schlussendlich auch in OpenAPI verfügbar. Das Backend gibt nicht direkt die Entitäten nach aussen, sondern es werden DTOs verwendet.

4.2.5.4 Konfiguration

Alles, was sich je Umgebung unterscheidet (Prod oder Lokal) wird ausserhalb des Codes festgehalten. Zugangsdaten und Schlüssel werden nie eingecheckt in die Versionsverwaltung, sondern werden zur Laufzeit bereitgestellt. Für alle Umgebungen hat man somit dasselbe Build-Artefakt, nur die Konfiguration ändert sich.

4.2.5.5 Validierung

Im Frontend gibt es Validierungen, welche der Benutzerführung dienen. Verbindlich ist schlussendlich jedoch immer das Backend. Der Controller prüft das Format und die Pflichtfelder, der Service die fachlichen Regeln und die Datenbank hat auch noch Absicherungen mit NOT NULL Constraints oder Fremdschlüssel Checks.

4.2.6 Zusammenfassung der Leitplanken

Diese Tabelle ist eine Zusammenfassung der Entscheidungen und dient als Vorlage für die technischen Entscheidungen sowie weitere Konzeptionen.

Die Leitplanken L13-L21 wurden rückwirkend aus dem weiteren Kapitel hier ergänzt, sodass man eine übersichtliche Liste hat.

Tabelle 22: Leitplanken

Leitplanken ID	Beschreibung
L1	Relationale Datenbank, jede Entität mit eigenem Primärschlüssel und Beziehungen über Fremdschlüssel (E1).
L2	Alle Daten eines Benutzers sind über Fremdschlüsselketten vollständig löschbar (E1).
L3	Die Schichten Controller → Service → Persistenz sind einzuhalten, ohne Schichten zu überspringen (E2).
L4	Das Backend ist fachlich paketiert. Zugriffe auf andere Bereiche nur über deren Service (E2).
L5	Die SPA enthält keine Geschäftslogik, ist in einer typisierten Sprache geschrieben und Mobile-First (E3).
L6	Autorisierung und verbindliche Validierung geschehen nur serverseitig (E3).
L7	Alle Bestandteile laufen als Container auf einem Host. Zugriff nur über den Reverse Proxy mit TLS (E4).
L8	Backend und Datenbank sind von aussen nicht erreichbar (E4).
L9	Das Deployment ist reproduzierbar, Zugangsdaten kommen zur Laufzeit (E4, Konfiguration).
L10	Einheitliches Fehlerformat ohne interne Details (Fehlerbehandlung).
L11	Logs ohne sensible Daten, ausgegeben auf die Standardausgabe (Logging).
L12	Versionierte REST/JSON-API mit dokumentiertem Vertrag und DTOs statt Entitäten (API-Stil).
L13	Backend mit Spring Boot 4 und Java 25, Frontend mit React 19, TypeScript (strict) und MUI. Die Versionen sind fixiert (E5, E6).
L14	Betrieb auf einem Infomaniak VPS Lite in der Schweiz mit Docker Compose, einziger Einstieg über Reverse Proxy Caddy, die Firewall lässt nur Ports 22/80/443 zu (E7).
L15	Authentifizierung über JWT im HttpOnly-Cookie, 8 Stunden aktiv
L16	Passwörter mit Argon2id und mindestens 15 Zeichen
L17	Jeder Datenzugriff prüft den Besitz anhand der Benutzer-ID aus dem Token.
L18	Eine Kontolöschung löscht alle Benutzerdaten hart
L19	Keine Drittanbieter-Dienste im Browser
L20	Automatisiert werden nur Unit-Tests geschrieben, rote Tests blockieren Merge und Deployment.
L21	Main Branch ist geschützt und kann nur über Pull Request mit erfolgreicher Pipeline geändert werden. Merge führt automatisches Deployment aus auf dem Server.

4.3 Technologie-Evaluation

Gemäss der Definition of Done wird in diesem Kapitel der finale Technologie-Stack geprüft und festgelegt. Als Grundlage dafür dienen die Leitplanken aus dem vorherigen Kapitel und die Vorkenntnisse.

Für die Entscheidungen wird die gleiche Methode wie bei der Systemarchitektur benutzt. Die Architekturtreiber werden weiterhin benutzt als Kriterien. Die Entscheidungsmatrix wird nur dann angewendet, wo es auch echte Alternativen gibt.

4.3.1 Entscheidungen

Diese Entscheidungen sind ausschlaggebend für die Umsetzung und man hat echte Alternativen. Darum kommt hier wieder die Entscheidungsmatrix zum Zug. Die Nummerierung setzt E1-E4 aus der Systemarchitektur fort.

4.3.1.1 E5 – Backend-Framework

Mit welchem Framework wird der Schichtenmonolith umgesetzt? Es gibt mittlerweile eine ganze Menge an verschiedensten Frameworks, welche alle ihre Daseinsberechtigung sowie Vor- und Nachteile haben.

Kandidaten:

- **Spring Boot**: Open-Source Java-Framework. Erkennt automatisch die benötigten Bibliotheken und verwaltet die Standardeinstellungen. Mit einem eingebetteten Tomcat-Webserver entsteht eine sofort ausführbare JAR-Datei.
- **Jakarta EE auf WildFly**: standardisierte Java-APIs (JAX-RS, JPA, CDI). Betrieb auf einem WildFly Applikationsserver
- **Quarkus**: auf Container optimiertes Java-Framework (schneller Start, wenig Speicher), ebenfalls auf Basis der Jakarta-APIs

Matrix:

Tabelle 23: Entscheidungsmatrix Backend-Framework

Kriterium	Gewicht	Spring Boot	Jakarta EE (WildFly)	Quarkus
T3 (Betrieb durch eine Person)	2	6 (12)	3 (6)	6 (12)
T4 (Zeitbudget)	2	6 (12)	3 (6)	3 (6)
T5 (Nutzung der vorhandenen Kernkompetenzen)	2	6 (12)	3 (6)	1 (2)
T7 (Serverseitige Authentifizierung und Autorisierung)	2	6 (12)	3 (6)	6 (12)
Total		48	24	32

Nicht bewertete Kriterien:

- T1 Datenhoheit: hängt am Datenmodell (E1), nicht am Framework
- T2 eine Codebasis und T9 Bedienung: Frontend-Themen
- T6 Erweiterbarkeit: bei allen drei gleichwertig
- T8 Testbarkeit: Unit-Tests mit JUnit und Mockito sind bei allen drei gleich möglich

Spring Boot sowie Quarkus laufen als eigenständige Anwendung in einem Container. Mit einem WildFly hat man einen zusätzlichen Applikationsserver, der konfiguriert und betrieben werden muss was zusätzlichen Aufwand bedeutet. Daher haben beim ersten Kriterium (T3) jeweils Spring Boot und Quarkus die Nase vorne.

Für Spring Boot sind schon wesentliche Vorkenntnisse vorhanden, mit dem Jakarta EE sowie WildFly nur Grundkenntnisse und beim Quarkus gar keine Kenntnisse vorhanden. Somit ist klar, dass für das Zeitbudget und Nutzungen der vorhandenen Kenntnisse Spring Boot klar vorne liegt.

Spring Security bringt auch Out of the Box Lösungen für Login, Passwort-Hashing und JWT-Unterstützung. In den Risiken wurde klar festgelegt, dass die Security an ein etabliertes Framework ausgelagert werden soll.

Für diese Diplomarbeit wird **Spring Boot** gewählt.

- Als konkrete Version wird Spring Boot 4 gewählt, welches der neuste Release ist auf Maven Central [7]
- Java 25 ist die aktuelle LTS-Version und wird von Spring Boot 4 voll unterstützt. Der Build wird mit Maven ausgeführt.
- Die API-Versionierung kann über die eingebaute Lösung vom Spring Framework 7 direkt benutzt werden. [8]

Bewusst in Kauf genommene Nachteile:

- Spring Boot 4 ist noch neu und man muss aufpassen, dass man bei Anleitungen im Internet auch die neusten Versionen liest.
- Die Spring Boot Lösung wird mehr Arbeitsspeicher brauchen als Quarkus. Beim Deployment ist dies zu beachten auf dem Server.

4.3.1.2 E6 – Frontend Framework

Es wurde entschieden, dass man eine Single-Page-Application (E3) umsetzt in einer typisierten Sprache (L5). Hier gibt es wie beim Backend-Framework auch eine ganze Menge an aktuellen Möglichkeiten. Daher hilft eine Entscheidungsmatrix sehr.

Kandidaten:

- **React:** Bibliothek für komponentenbasierte Oberflächen. Routing, Datenabfragen und Formulare werden über ergänzende Bibliotheken gehandhabt.
- **Angular:** vollständiges Framework mit Routing, Formularen, HTTP-Client und TypeScript als Standard.
- **Vue:** progressives Framework mit einfacher Einarbeitung und TypeScript-Unterstützung.

Matrix:

Tabelle 24: Entscheidungsmatrix Frontend-Framework

Kriterium	Gewicht	React	Angular	Vue
T4 (Zeitbudget)	2	6 (12)	1 (2)	3 (6)
T5 (Nutzung der vorhandenen Kernkompetenzen)	2	3 (6)	1 (2)	1 (2)
T6 (Erweiterbarkeit)	1	6 (6)	3 (3)	3 (3)
Total		24	7	11

Nicht bewertete Kriterien:

- T2 eine Codebasis und T9 Bedienung: ist bei allen drei gleichwertig. Sie sind alle responsive und genügend schnell für die Liveprotokollierung.
- T8 Testbarkeit: alle drei sind Unittests tauglich
- T1, T3 und T7: Keine Frontend Themen

Die Grundkenntnisse in React und TypeScript zusammen mit dem grössten Angebot an Beispielen führen zu einem hohen Total für React. Angular hat am Anfang eine steile Lernkurve, die für diese Diplomarbeit schlichtweg nicht vertretbar ist.

React hat ein riesiges Ökosystem mit jeglichen Komponentenbibliotheken, die verwendet werden können.

Für diese Diplomarbeit wird **React mit TypeScript** gewählt.

- Die aktuelle Version, welche benutzt wird, ist React 19 [9]
- Für TypeScript wird der Release 6 verwendet im «strict» Modus. Somit ist der Schnittstellenvertrag gegeben. Der neuste Release TypeScript 7 unterstützt noch kein ESLint. [10]
- Das Build-Werkzeug, welches verwendet wird, ist Vite
- Node.js 24 LTS als Build-Umgebung

Bewusst in Kauf genommene Nachteile:

- React ist nur eine Bibliothek. Für das Routing, Datenabfragen und Formulare wird es noch weitere Entscheidungen brauchen.
- Jede zusätzliche Bibliothek erhöht die Unübersichtlichkeit des ganzen Projekts.

4.3.1.3 E7 – Hosting Provider

Bei welchem Anbieter wird der einzelne Server (E4) betrieben?

Kandidaten (Preise Stand 07/2026):

- **DigitalOcean Droplet:** US-Anbieter mit einem Rechenzentrum in Frankfurt. Kosten ca. CHF 10 pro Monat für einen Server mit mindestens 2 GiB RAM.
- **Hetzner Cloud:** Deutscher Anbieter mit Rechenzentren in Deutschland. DSGVO-konforme Cloud-Server ab CHF 6 pro Monat.
- **Infomaniak VPS Lite:** Schweizer Anbieter mit einem Rechenzentrum in der Schweiz. Server mit 4 GB RAM kostet mindestens CHF 7 pro Monat.

Matrix:

Tabelle 25: Entscheidungsmatrix Hosting Provider

Kriterium	Gewicht	DigitalOcean	Hetzner	Infomaniak
T1 (Datenhoheit inkl. Löschbarkeit)	2	1 (2)	3 (6)	6 (12)
T3 (Betrieb durch eine Person, 10–15 CHF/Monat)	2	3 (6)	6 (12)	6 (12)
T4 (Zeitbudget)	2	6 (12)	3 (6)	3 (6)
Total		20	24	30

Nicht bewertete Kriterien:

- T2, T5, T7, T8 und T9: unabhängig vom Anbieter
- T6 Erweiterbarkeit: alle drei bieten Objektspeicher für das Kann-Ziel Medien-Upload

Die Datenhoheit spielt für den Hosting Provider eine hohe Rolle. Da DigitalOcean ein US-Unternehmen ist und somit die US-Behörden theoretisch Zugriff auf die Daten haben ist es aus Datenschutzgründen nicht vertretbar. Bei Hetzner mit deutschen Rechenzentren ist das schon besser. Deutlicher Gewinner in diesem Kriterium ist jedoch Infomaniak, welches die lokalen Schweizer Datenschutzrichtlinien einhalten muss.

Alle Hosting-Provider liegen im Budget. DigitalOcean ist ein grosser Player und hat dadurch auch eine sehr umfangreiche Anleitungssammlung und vorkonfigurierte Docker-Images was für eine schnelle Einrichtung sorgt. Jedoch ist das auch auf Hetzner oder Infomaniak keine grosse Hexerei heutzutage.

Für diese Diplomarbeit wird **Infomaniak VPS Lite** gewählt.

- Server mit 2 vCPU, 4 GB RAM und 60 GB Festplattenspeicher
- Rechenzentrum liegt in der Schweiz und unterliegt dem DSG
- Die Anbieterauswahl für das Kapitel Deployment ist somit vorgezogen. Die Einrichtung des Servers erfolgt zu einem späteren Zeitpunkt.

Bewusst in Kauf genommene Nachteile:

- Wie schon erwähnt würde zum Beispiel mit DigitalOcean das Einrichten von Docker einfacher sein.
- Nicht die günstigste Lösung, Hetzner wäre pro Monat minim günstiger.

4.3.2 Ergänzende Technologien

Mit den Entscheidungen über das Backend-Framework, Frontend Framework und dem Hosting-Provider hat man eine solide Basis der Technologien. Um den Technologiestack aber zu vervollständigen, braucht es noch einige ergänzende Technologien.

Backend

Tabelle 26: Ergänzende Technologien Backend

Bereich	Technologie	Begründung
Sprache und Build	Java 25 (LTS), Maven	Die sehr hohe Erfahrung mit Java und Maven spricht für sich bei dieser Wahl. Mit dem POM File können die Versionen klar definiert werden.
Datenbank	PostgreSQL 18	Aufgrund der Vorkenntnisse wird die aktuelle Version von PostgreSQL benutzt als Datenbank.
Persistenz	Spring Data JPA (Hibernate)	Es gibt verschiedenste Umsetzungen für die Persistenz innerhalb der Spring-Boot-Applikation. Spring Data JPA bietet alle modernen Möglichkeiten und ist gut dokumentiert.
Validierung	Jakarta Bean Validation	Hiermit kann man das Format und die Pflichtfelder im Controller sauber validieren.
REST-Controller	Spring Web	Diese Spring Boot Dependency macht die App erst zu einem Webserver der REST-Calls empfangen und senden kann.
Security	Spring Security, OAuth2 Resource Server	Dieser Ansatz bietet ein Login mit Authentifizierung und Autorisierung ohne eigene Kryptografie.
API-Dokumentation	springdoc-openapi 3	Damit lassen sich direkt aus dem Code die OpenAPI-Spezifikation und Swagger UI erstellen.
DTO-Mapping	Java Records + eigene Mapper	keine Zusatzbibliothek dafür nötig
Migration	Flyway	Versionierte Datenbankänderungen.
Health-Check	Spring Boot Actuator	Überwachungstool.
Tests	JUnit, Mockito, AssertJ	Im Test-Starter von Spring Boot enthalten

Frontend

Tabelle 27: Ergänzende Technologien Frontend

Bereich	Technologie	Begründung
UI-Komponenten	Material UI	Fertige Bausteine wie Buttons, Eingabefelder und Dialoge.
Seitenwechsel	React Router	Zeigt je nach URL die passende Ansicht, ohne dass die Seite neu lädt.
Anfragen ans Backend	Fetch	Schickt die Anfragen an die REST-Schnittstelle ohne Zusatzbibliothek.
Formulare	React Hook Form	Wickelt die Eingabefelder ab und zeigt Fehlermeldungen direkt beim Feld an.
Code-Qualität	ESLint, Prettier	Prettier formatiert den Code einheitlich, ESLint findet Fehler.
Tests	Vitest, React Testing Library	Passende Testwerkzeuge.

Infrastruktur und Werkzeuge

Tabelle 28: Ergänzende Technologien Infrastruktur

Bereich	Technologie	Begründung
Container	Docker, Docker Compose	Benutzung von Docker wurde schon festgelegt.
Betriebssystem	Ubuntu LTS	Standard-Linux für einen Cloud Server.
Reverse Proxy	Caddy	Bietet automatische HTTPS Zertifizierung und eine kurze übersichtliche Konfiguration. Dient als einziger Einstiegspunkt.
Firewall	Firewall von Infomaniak	Nur die nötigen Ports öffnen.
Versionsverwaltung	Git, GitHub	Starke Vorkenntnisse und absolut nötig für jedes Softwareentwicklungsprojekt.
CI/CD	GitHub Actions	Direkt integriert in die Versionsverwaltung.

4.3.3 Übersicht Technologie-Stack

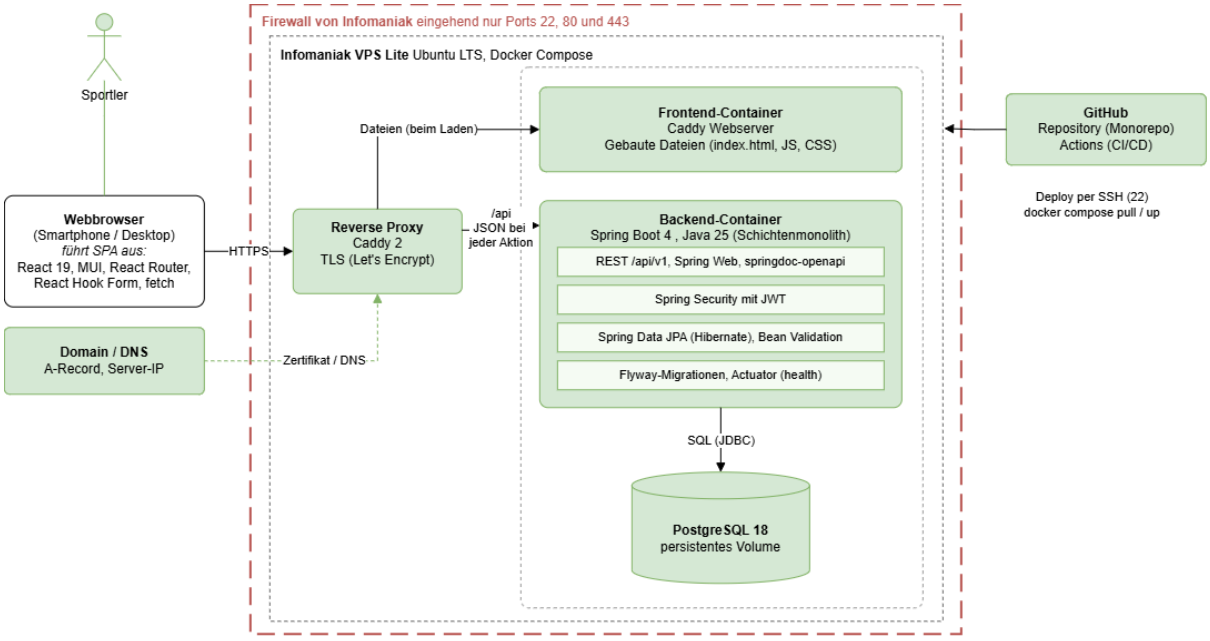


Abbildung 6: Technologie-Stack

4.4 Cyber Security Konzept

In diesem Kapitel wird die Sicherheitsarchitektur auf Basis des Technologie-Stacks festgelegt. Die Schutzmassnahmen werden anhand der aktuellen OWASP-Schwachstellen definiert [11]. Zentral dabei ist, dass es keine eigene Krypto oder Session-Mechanismen gibt, sondern man dafür etablierte Standards einsetzt.

4.4.1 Schutzmassnahmen nach OWASP TOP 10:2025

Tabelle 29: Schutzmassnahmen nach OWASP

ID	Risiko	Massnahmen
A01:2025	Broken Access Control	Deny by default. Backend, Frontend und DB von aussen nicht erreichbar. Jeder Datenzugriff prüft den Besitz anhand der Benutzer-ID aus dem Token.
A02:2025	Security Misconfiguration	Secrets nur über Umgebungsvariablen und nicht hardcodiert im Code. Keine Stacktraces nach aussen. Swagger UI nur lokal
A03:2025	Software Supply Chain Failures	So wenig Abhängigkeiten wie möglich. Fixierte Versionen benutzen.
A04:2025	Cryptographic Failures	Keine eigene Kryptografie verwenden. Bestehende Frameworks benutzen.
A05:2025	Injection	Spring Data JPA mit Parametern gegen SQL Injection.
A06:2025	Insecure Design	Sicherheitskonzept vor der Entwicklung festlegen und daran festhalten.
A07:2025	Authentication Failures	Passwort-Richtlinien, generische Fehlermeldungen und ein Token-Ablauf
A08:2025	Software or Data Integrity Failures	Images werden nur in der Pipeline gebaut. Deployment auch nur über die Pipeline.
A09:2025	Security Logging and Alerting Failures	Beim Login Fehlversuche und verweigerte Zugriffe loggen.
A10:2025	Mishandling of Exceptional Conditions	Zentrale Fehlerbehandlung mit einem einheitlichen Fehlerformat. Transaktionen mit Rollback.

4.4.2 Schutzobjekte und Schutzbedarf

Tabelle 30: Schutzobjekte und Schutzbedarf

Schutzobjekt	Beispiele	Schutzbedarf	Begründung
Zugangsdaten	E-Mail, Passwort-Hash	hoch	Kontoübernahme, Passwörter werden oft mehrfach verwendet
Trainingsdaten	Übungen, Routinen, Einheiten, Sätze, Gewichte, Notizen	mittel	Personendaten, aber nicht besonders schützenswert.
Secrets	JWT-Schlüssel, DB-Passwort, SSH-Schlüssel	hoch	Werden diese Daten kompromittiert hat ein Angreifer Zugriff auf alle Daten
Verfügbarkeit	Container	niedrig	In der Deployment-Topologie (E4) wurde ein kurzer Ausfall des Hosts akzeptiert.

4.4.3 Sicherheitszonen

- **Firewall:** Firewall von Infomaniak. Nur die eingehenden Ports 22 (SSH), 80 und 443 sind offen. Der Rest wird blockiert.
- **Reverse Proxy:** Der Caddy Reverse Proxy dient als einziger Einstiegspunkt. TLS-Terminierung und Umleitung auf HTTPS
- **Applikationen:** Frontend und Backend Container sind im internen Docker-Netz ohne veröffentlichte Ports.
- **Daten:** PostgreSQL ist nur vom Backend erreichbar.

4.4.4 Authentifizierungsverfahren

Nach dem Login muss sich der Benutzer bei jeder weiteren Anfrage ausweisen können. Dafür wird ein JSON Web Token (JWT) verwendet. Das Backend stellt es nach erfolgreichem Login selbst aus. Das Erstellen und Prüfen übernimmt Spring Security mit dem OAuth2 Resource Server. So braucht es keine eigene Kryptografie, wie es in der Massnahme zu R05 festgehalten ist.

Das Token ist «stateless». Der Server merkt sich also nicht die Sitzung, sondern prüft bei jeder Anfrage nur die Signatur und das Ablaufdatum des Tokens. Ein Neustart des Backends, zum Beispiel nach einem Deployment meldet niemanden ab. Somit können Live-Protokollierungen weiter ohne neues Anmelden durchgeführt werden.

Das Token beinhaltet Benutzer-ID sowie den Ausstellungs und Ablaufzeitpunkt. Informationen wie E-Mail-Adresse oder Name stehen nicht drin. Das Token wird signiert mit HS256 und einem zufälligen Schlüssel von mindestens 256 Bit. Dieser Schlüssel wird über eine Umgebungsvariable bereitgestellt. Ein Token ist 8 Stunden gültig. Diese Zeit sollte ein normales Training abdecken. Anschliessend ist ein neues Login nötig. Zusätzlich prüft das Backend, ob der Benutzer noch existiert bei jeder Abfrage. Einem gelöschten Konto wird sofort der Zugriff verwehrt.

Das Token wird also nicht auf dem Backend sondern im Browser gespeichert. Dies in einem HttpOnly-Cookie. JavaScript des Frontends darf das Cookie nicht lesen. Wenn fremder Code in die Seite gelangen würde, könnte er das Token gar nicht stehlen. Das Cookie wird nur über HTTPS übertragen. Das Cookie enthält zusätzlich «SameSite=Strict». Somit ist man vor CSRF besser geschützt.

4.4.5 Autorisierung

Es wird das «Deny by default» Verfahren angewendet. Alle Endpunkte brauchen ein gültiges Token. Einzige Ausnahmen sind die Registrierung, Login und Health-Check.

Es gibt im Backend immer eine Besitzerprüfung anhand der Benutzer-ID aus dem Token. Man hat keinen Zugriff auf fremde Daten anderer IDs. Die ID darf nicht aus der URL oder dem Request Body übernommen werden.

Greift man auf Fremde oder nicht vorhandene Datensätze zu, erhält man einen einfachen 404 Errorcode und es wird nicht verraten, ob ein Datensatz existiert.

Es ist kein CORS nötig, da das Frontend und die API über dieselbe Domain laufen.

4.4.6 Passwörter und Login

Das Passwort wird mit Argon2id gehasht (erste Wahl gemäss OWASP) [12]. Im Spring Boot wird das mit dem Argon2PasswordEncoder umgesetzt. Da es keine 2FA geben wird braucht es strikte Passwort-Richtlinien. Diese sind:

- **Länge:** Mindestens 15 Zeichen, Maximal 64 Zeichen.
- **Zusammensetzung:** Es wird nicht vorgeschrieben, aus welchen Zeichen das Passwort bestehen muss.
- **Periodischer Wechsel:** Man wird nicht gezwungen alle 30, 60 oder 90 Tage das Passwort zu ändern.
- **Einfügen und Passwort-Manager erlaubt:** Das Passwort kann aus der Zwischenablage kopiert werden. Das fördert die Nutzung von Passwort-Managern. Das Passwort kann zur Prüfung dann auch angezeigt werden.

Wenn ein Login fehlschlägt, gibt es nur eine generische Fehlermeldung und keinen Hinweis darauf, ob es den Benutzer oder das Passwort so gibt. Nach 5 Fehlversuchen wird das Konto für 15 Minuten gesperrt. Dies schützt gegen Brute-Force-Attacken.

Es wird kein Passwort-Reset angeboten, da es dafür einen Mailserver bräuchte.

4.5 Datenschutz und rechtliche Aspekte

Der Datenschutz muss berücksichtigt werden. Dieses Kapitel hält fest, welche Personendaten die Applikation bearbeitet und wie die Vorgaben des Schweizer Datenschutzgesetzes umgesetzt werden.

Massgebend ist das Bundesgesetz über den Datenschutz, welches seit dem 1. September 2023 in Kraft ist [13]. Wenn eine Person Daten ausschliesslich zum persönlichen Gebrauch bearbeitet, gilt das DSG nicht. Da der Server Öffentlich ist und die Registrierung von anderen Personen gemacht werden kann wird die Applikation von Beginn an DSG-konform umgesetzt. So braucht es später keine Nacharbeit.

Verantwortlich für die Datenbearbeitung ist Lennard Bernet als Betreiber der Applikation.

Infomaniak bearbeitet die Daten als Hosting-Provider im Auftrag von Lennard Bernet und unterliegt dem DSG.

4.5.1 Bearbeitende Personendaten

Es werden nur die Daten erhoben, welche für die Trainingsverwaltung auch wirklich nötig sind.

Tabelle 31: Personendaten

Kategorie	Daten	Zweck	Aufbewahrung
Konto	E-Mail-Adresse, Passwort-Hash, Persönliche Angaben (Vorname, Nachname)	Anmeldung, Kontoverwaltung	bis zur Kontolöschung
Trainingsdaten	Übungen, Routinen, Planungen, Trainingseinheiten, Sätze	Kernfunktion Trainingsverwaltung	bis zur Kontolöschung durch den Benutzer
Technische Daten	IP-Adresse, Zeitstempel, Sicherheitsereignisse	Betrieb und Sicherheit	Kurz, nur in den Logfiles
Cookie	Token-Cookie (HttpOnly)	Anmeldung	max. 8 Stunden

Gesundheitsdaten gelten nach Art. 5 DSG als besonders schützenswert. Trainingsdaten wie Gewichte oder Wiederholungen beschreiben eine Trainingsleistung und keinen Gesundheitszustand. Damit das so bleibt, gibt es bewusst keine Felder für Körpergewicht, Körpermasse oder Verletzungen. [14] [15]

4.5.2 Umsetzung der Grundsätze

- **Datenminimierung:** Für die Registrierung reichen E-Mail und Passwort. Es gibt kein Tracking, keine Analyse-Tools und keine Werbung.
- **Zweckbindung:** Die Daten werden nur für die Trainingsverwaltung verwendet und nicht an Dritte weitergegeben.
- **Speicherort Schweiz:** Infomaniak ist ein Schweizer Unternehmen und betreibt den Server in der Schweiz. Somit gibt es keine Übermittlung der Daten ins Ausland.
- **Datensicherheit:** Die technischen Massnahmen sind im Cyber Security Konzept beschrieben.

4.5.3 Löschkonzept

Jeder Benutzer kann sein Konto jederzeit selbst löschen im Profil. Zur Bestätigung der Löschung muss er nochmals sein Passwort eingeben. Anschliessend werden in einer einzigen Transaktion alle abhängigen Daten hart gelöscht auf der Datenbank. Es gibt also kein «Soft Delete» bei der Löschung des Benutzers und auch keine Archivierung. Hat der Benutzer ein noch gültiges Token, wird dieses durch die Benutzerprüfung sofort wertlos. In den Logs sind nur die IDs zu dem Benutzer enthalten, welche dann bei der folgenden Log-Rotation komplett verschwinden.

Wenn der Benutzer einfach eine Trainingseinheit, Übung oder Routine löscht, kann ein «Soft Delete» angewendet werden, je nach Fall. Es wird ein Flag auf dem Datensatz «Deleted» gesetzt und nicht mehr angezeigt. Somit kann auf spezielle Anfragen diese Löschung rückgängig gemacht werden.

Vergisst der Benutzer sein Passwort, kann er die Löschung per E-Mail beim Verantwortlichen beantragen.

4.5.4 Rechte der Benutzer und Datenschutzerklärung

Die Rechte der betroffenen Personen sind übersichtlich und werden kurzgehalten:

- **Auskunft und Datenherausgabe:** Auf Anfrage werden die Daten manuell als JSON exportiert.
- **Berichtigung:** Der Benutzer kann in seinem Profil seine eigenen Daten bearbeiten und löschen.
- **Löschung:** Gemäss Löschkonzept.

Es wird eine eigene Seite für den Datenschutz geben. Diese informiert über den Verantwortlichen, den Zweck, die bearbeiteten Daten, den Speicherort und die Aufbewahrung. Diese Seite ist vor der Registrierung erreichbar.

4.6 Teststrategie

Gemäss Definition of Done muss eine Teststrategie für Backend und Frontend konzeptionell dokumentiert sein. Man will die Geschäftslogik absichern und die Fehler möglichst früh finden. Daraus entstehen dann automatisierte Unit-Tests welche bewusst einfach gehalten werden. Integrationstests im Spring Kontext werden auch als Unit-Test im Backend markiert.

Testcontainer, End-to-End-Tests und Last- und Performancetests werden bewusst weggelassen. Dafür gibt es manuelle Systemtests anhand von Testfällen, welche aus den Anforderungen kommen sowie Smoke-Tests nach jedem Deployment.

Mit diesen Teststufen hat man eine Übersicht aller Tests, die vorgesehen sind. Die konkrete Spezifikation davon wird erst zum nötigen Zeitpunkt festgehalten. Dies dient nur als Grundlage was getestet werden soll mit welchem Werkzeug.

Tabelle 32: Teststufen

Teststufe	Prüfgegenstand	Werkzeug / Form	Zeitpunkt
Unit-Test Backend	Service-Schicht (Geschäftsregeln), Mapper, Hilfsklassen, Erstellen und Prüfen der Tokens	JUnit, Mockito, AssertJ	Entwicklung per TDD, automatisch in der Pipeline
Unit-Test Frontend	reine Funktionen und zentrale Komponenten	Vitest, React Testing Library	Entwicklung, automatisch in der Pipeline
Manueller Systemtest	Ausgewählte Anwendungsfälle und Anforderungen.	Testprotokoll	Entwicklung
Smoke-Test	Login, Kernablauf, HTTPS	Manuell in Produktion	Nach Deployments
Review	Sichtprüfung von Code und Konfiguration.	Manuell im Code	Entwicklung

4.7 Versionsverwaltung

In den ergänzenden Technologien wurde Git und GitHub für die Versionsverwaltung festgelegt. Hier wird der Einsatz klar definiert und eine verbindliche Branching Strategie festgehalten. Das muss vor der Entwicklung festgelegt sein. Somit hat man vom Anfang an eine saubere Versionsverwaltung die nachvollziehbar ist.

Umgesetzt wird ein Monorepo für Backend, Frontend und Deployment-Konfiguration.

Die Grundstruktur ist folgendermassen aufgeteilt:

- /backend -> Spring Boot mit Maven
- /frontend -> React App
- /deploy -> Docker Konfigurationen, Reverse Proxy Konfiguration
- /.GitHub/workflows -> CI/CD Pipelines

Somit ist sichergestellt, dass wenn es Schnittstellenänderungen gibt, diese in einem einzelnen Pull Request abgehandelt werden und man eine einzige Pipeline hat. Für einen Einzelentwickler ist dieser Aufbau somit optimal.

Der main Branch ist jeweils der aktuelle Stand der Produktion. Ein Merge auf den Main Branch löst ein Deployment aus. Die Änderungen auf diesen Branch können nur per Pull Request mit einer grünen Pipeline gemacht werden.

Der Link zum Repository: https://GitHub.com/bernetlennard/digital-training-platform

4.8 CI/CD-Pipeline

Es werden die wesentlichen Werkzeuge für die CI/CD Pipelines definiert und ein grober Ablauf festgehalten. Das Grundgerüst entsteht früh was in der Risikomassnahme R02 definiert wurde.

4.8.1 Werkzeuge

Zusammenfassung der bestehenden Werkzeuge die definiert wurden.

Tabelle 33: Werkzeuge CI/CD

Zweck	Werkzeug	Begründung
Pipeline	GitHub Actions	Ist direkt im GitHub integriert
Build Backend	Maven, JDK 25	Gleiche Befehle wie lokal
Build Frontend	npm ci, Vite	Reproduzierbar
Container-Images	Multi-Stage-Dockerfiles	Minimale Image-Grösse ohne unnötige Abhängigkeiten.
Registry	Dockerhub	Es lassen sich direkt von GitHub aus die Container-Images speichern und verwalten.
Auslieferung	SSH + Docker Compose	Braucht keine zusätzliche Komponente auf dem Server.
Secrets	GitHub Actions Secrets	SSH-Schlüssel und Host, App-Secrets bleiben in der .env auf dem Server.

4.8.2 Umgebungen

- **Lokal:** Backend aus der IDE, Frontend über den Vite-Entwicklungsserver, PostgreSQL als Docker Container.
- **Produktion:** Infomaniak VPS Lite mit docker-compose.yml (Images für caddy, frontend, backend und db)
- **Keine Testumgebung:** Aus Kosten und Zeitgründen wird keine Testumgebung aufgesetzt.

4.8.3 Ablauf der Pipeline

- **Pull Request auf main (CI-Jobs)**
  - **Job Backend:** Kompiliert das Backend und führt die Unit-Tests durch
  - **Job Frontend:** ESLint Prüfung, TypeScript Typprüfung, Vitest-Tests, Vite Build
  - **Beide Jobs grün:** Merge erlaubt.
- **Merge auf Main**
  - CI-Jobs erneut ausführen.
  - Images für Backend und Frontend bauen und aufs Docker Repository pushen.
  - Deploy auf den Server per SSH. Image-Tag setzen und docker compose pull sowie docker compose up laufen lassen.
  - Das Datenbankschema wird beim Start des Backends automatisch aktualisiert.
  - Smoke-Check: Aufruf des Health-Endpunkts über HTTPS

4.9 UI/UX-Strategie

Da die Vorkenntnisse für den gesamten Frontend Teil nicht so gross sind hilft hier eine Komponentenbibliothek für das eigene Styling. Dafür wurde Material UI gewählt.

Folgende Gestaltungsgrundsätze müssen während der Konzeption und Entwicklung eingehalten werden.

- Mobile-First
- Touch-Ziele gut bedienbar
- Wenig selbst tippen. Zahlenfelder, Vorbelegung mit den Werten der letzten Einheit
- Gut lesbar
- Dunkelmodus ist optional und wird im ersten Entwurf nicht berücksichtigt.
- Klare Rückmeldungen bei Bestätigungen und Fehlern.
- Standardkomponenten aus Material UI verwenden statt eigenes CSS

5 Anforderungsanalyse

Das Ziel der Anforderungsanalyse ist es festzulegen, was das System fachlich leisten soll. Die gesamte Anforderungsanalyse besteht aus 6 Geschäftsanwendungsfällen, 9 Systemanwendungsfällen, 7 Features mit 42 funktionalen und 38 nicht-funktionalen Anforderungen. Es wurden ausschliesslich die Muss-Ziele abgeleitet.

Hier nochmals eine Übersicht der Muss-Ziele mit einer jeweiligen ID und Beschreibung:

Tabelle 34: Muss-Ziele

ID	Muss-Ziel
M1	Benutzerverwaltung mit Registrierung, serverseitigem Login und Löschkonzept
M2	Übungskatalog und Trainingsroutinen erstellen, bearbeiten und löschen
M3	Trainingsroutine für zukünftige Daten einplanen
M4	Historie einer Trainingsroutine einsehen
M5	Trainingseinheit live protokollieren: Sätze, Wiederholungen, Gewichte, Pausenzeiten, Muskelgruppe, Beschreibung
M6	Vollständig responsive Web-Applikation für Smartphone und Desktop

Als Akteure kommen folgende zum Zug:

- **Besucher:** Ein Benutzer, der auf der Plattform noch nicht registriert ist.
- **Sportler:** Ein Benutzer, der auf der Plattform registriert ist.
- **Betreiber:** Inhaber und Administrator der Plattform.

5.1 Geschäftsanwendungsfälle

Das Training wird in vier Kern-Geschäftsanwendungsfälle und zwei unterstützende Geschäftsfälle unterschieden. Diese wurden aus den Muss-Zielen, Projektvision und dem eigenen Ablauf im Studio entwickelt.

- **Kern-GAF:** Diese bringen einen direkten Nutzen für die Leistungssteigerung
- **Unterstützende GAF:** Diese sind die Voraussetzung für die Kern-GAF

Jeder GAF hat einen fachlichen Auslöser und ein daraus folgendes Ergebnis.

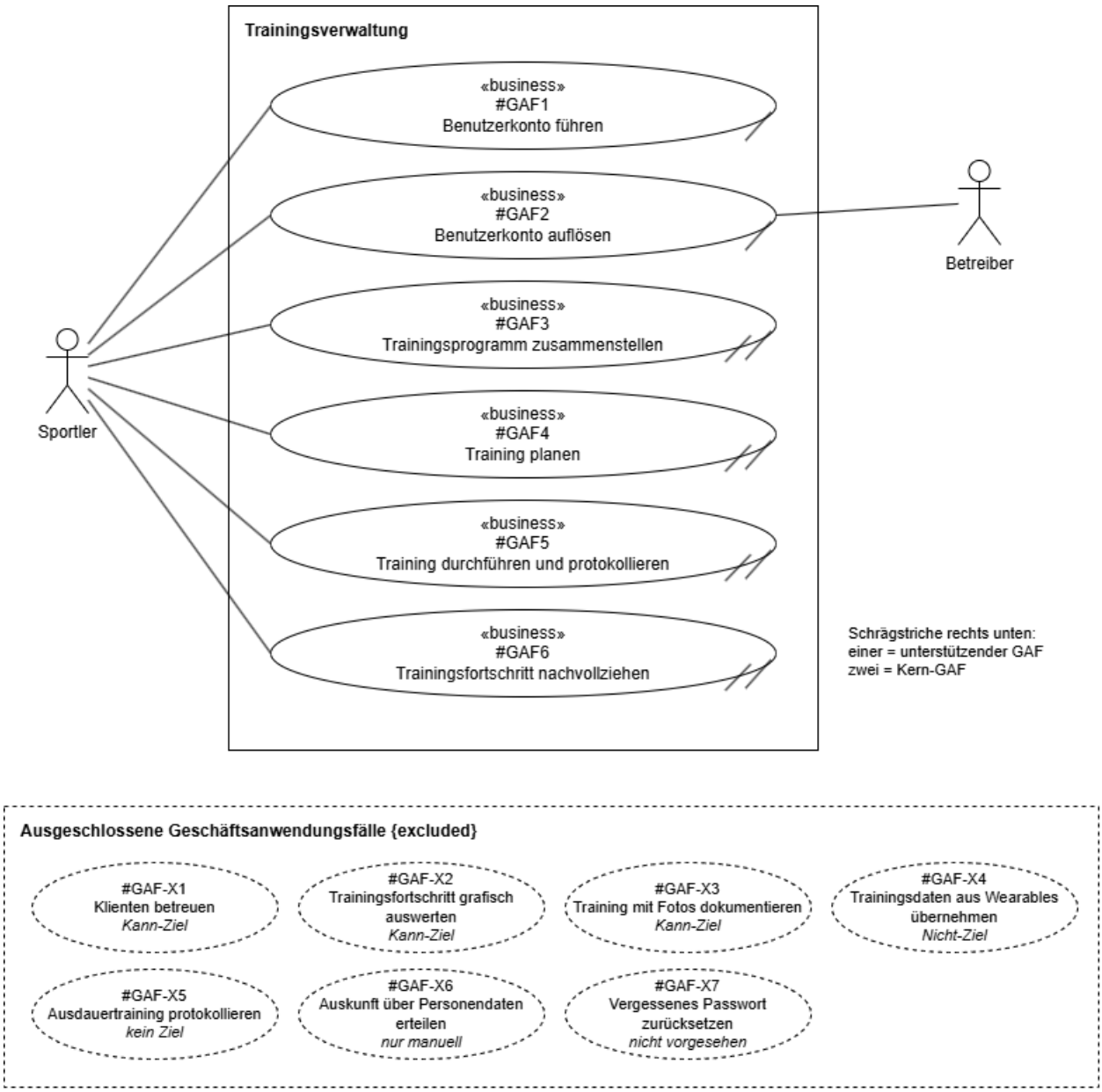


Abbildung 7: Geschäftsanwendungsfälle

5.1.1 GAF1 Benutzerkonto führen

Tabelle 35: GAF1 Benutzerkonto führen

Feld	Inhalt
Art	Geschäftsanwendungsfall, unterstützend
Kurzbeschreibung	Der Sportler kann auf der Plattform ein persönliches Konto führen, auf das nur er zugreifen kann und seine eigenen Konto- und Trainingsdaten verwaltet.
Auslöser	Der Sportler will seine persönlichen Trainings auf einer geschützten Plattform selbst verwalten.

Ergebnis	Auf der Plattform hat der Sportler nun sein eigenes Konto mit Zugriff ausschliesslich auf seine eigenen Daten
Akteure	Sportler

5.1.2 GAF2 Benutzerkonto auflösen

Tabelle 36: GAF2 Benutzerkonto auflösen

Feld	Inhalt
Art	Geschäftsanwendungsfall, unterstützend
Kurzbeschreibung	Laut den Datenschutzbestimmungen der Plattform hat man die Möglichkeit, sein eigenes Konto mit allen Trainingsdaten vollständig und endgültig zu löschen. Entweder macht der Sportler das direkt im Profil oder per Anfrage an den Betreiber, wenn man das Passwort vergessen hat.
Auslöser	Der Sportler will die Plattform nicht mehr nutzen und seine Daten entfernen.
Ergebnis	Konto und sämtliche Trainingsdaten sind endgültig entfernt.
Akteure	Sportler, Betreiber

5.1.3 GAF3 Trainingsprogramm zusammenstellen

Tabelle 37: GAF3 Trainingsprogramm zusammenstellen

Feld	Inhalt
Art	Geschäftsanwendungsfall, Kern
Kurzbeschreibung	Der Sportler legt seine Übungen mit Muskelgruppe und Beschreibung fest und stellt daraus Trainingsroutinen in fester Reihenfolge zusammen
Auslöser	Der Sportler beginnt ein neues Trainingsprogramm oder passt sein bestehendes an.
Ergebnis	Einsatzbereite Trainingsroutinen.
Akteure	Sportler

5.1.4 GAF4 Training planen

Tabelle 38: GAF4 Training planen

Feld	Inhalt
Art	Geschäftsanwendungsfall, Kern
Kurzbeschreibung	Der Sportler plant seine nächsten Trainings mit den jeweiligen Routinen ein. Bestehende Planung soll auch angepasst werden können.
Auslöser	Es fängt eine neue Woche an und der Sportler will seinen Trainingsplan auf der Plattform hinterlegen.
Ergebnis	Für die gesamte Woche ist festgelegt, an welchem Tag mit welcher Routine und Übungen trainiert wird.
Akteure	Sportler

5.1.5 GAF5 Training durchführen und protokollieren

Tabelle 39: GAF5 Training durchführen und protokollieren

Feld	Inhalt
Art	Geschäftsanwendungsfall, Kern
Kurzbeschreibung	Der Sportler ist im Fitnesscenter und absolviert eine geplante Trainingseinheit. Pro Übung hält er die Sätze, Gewicht, Wiederholungen und Pausenzeiten fest.
Auslöser	Der Sportler beginnt im Studio ein geplantes oder spontanes Training.
Ergebnis	Eine vollständig dokumentierte Trainingseinheit.
Akteure	Sportler

5.1.6 GAF6 Trainingsfortschritt nachvollziehen

Tabelle 40: GAF6 Trainingsfortschritt nachvollziehen

Feld	Inhalt
Art	Geschäftsanwendungsfall, Kern
Kurzbeschreibung	Der Sportler will nachschauen, wie seine Leistung sich bei den gleichen Trainingsroutinen verändert hat. Daraus kann er feststellen, ob ein progressiver Fortschritt vorhanden ist.
Auslöser	Der Sportler will wissen, welche Gewichte und Wiederholungen er in den letzten Trainingseinheiten geschafft hat.
Ergebnis	Der Sportler hat eine Übersicht seiner letzten Trainingseinheiten erhalten und weiss, wo er die Gewichte erhöhen muss.
Akteure	Sportler

5.1.7 Ausgeschlossene Geschäftsanwendungsfälle

Tabelle 41: Ausgeschlossene GAF

ID	Name	Inhalt	Grund
#GAF-X1	Klienten betreuen	Trainer plant und überwacht Trainings seiner Klienten	Kann-Ziel «Rollen mit Kundenverwaltung»
#GAF-X2	Trainingsfortschritt grafisch auswerten	Verlaufscharts, Statistiken, Dashboards	Kann-Ziel «visuelle Aufbereitung» GAF6 umfasst nur die Einsicht
#GAF-X3	Training mit Fotos dokumentieren	Bilder an Übungen oder Einheiten	Kann-Ziel «Medien-Upload»
#GAF-X4	Trainingsdaten aus Wearables übernehmen	Puls, GPS, Distanz von Smartwatches	Nicht-Ziel
#GAF-X5	Ausdauertraining protokollieren	Zeit, Distanz, Tempo	Fokus progressives Krafttraining
#GAF-X6	Auskunft über Personendaten erteilen	Sportler kann direkt auf der Plattform seine Daten als JSON exportieren.	Wird höchstens manuell erfüllt auf Anfrage beim Betreiber.
#GAF-X7	Vergessenes Passwort zurücksetzen	Neues Passwort per E-Mail-Link	Kein Passwort-Reset und kein Mailversand. Profil kann auf Anfrage gelöscht werden.

5.2 Systemanwendungsfälle

Die sechs relevanten Geschäftsanwendungsfälle werden nun in neun Systemanwendungsfälle zerlegt.

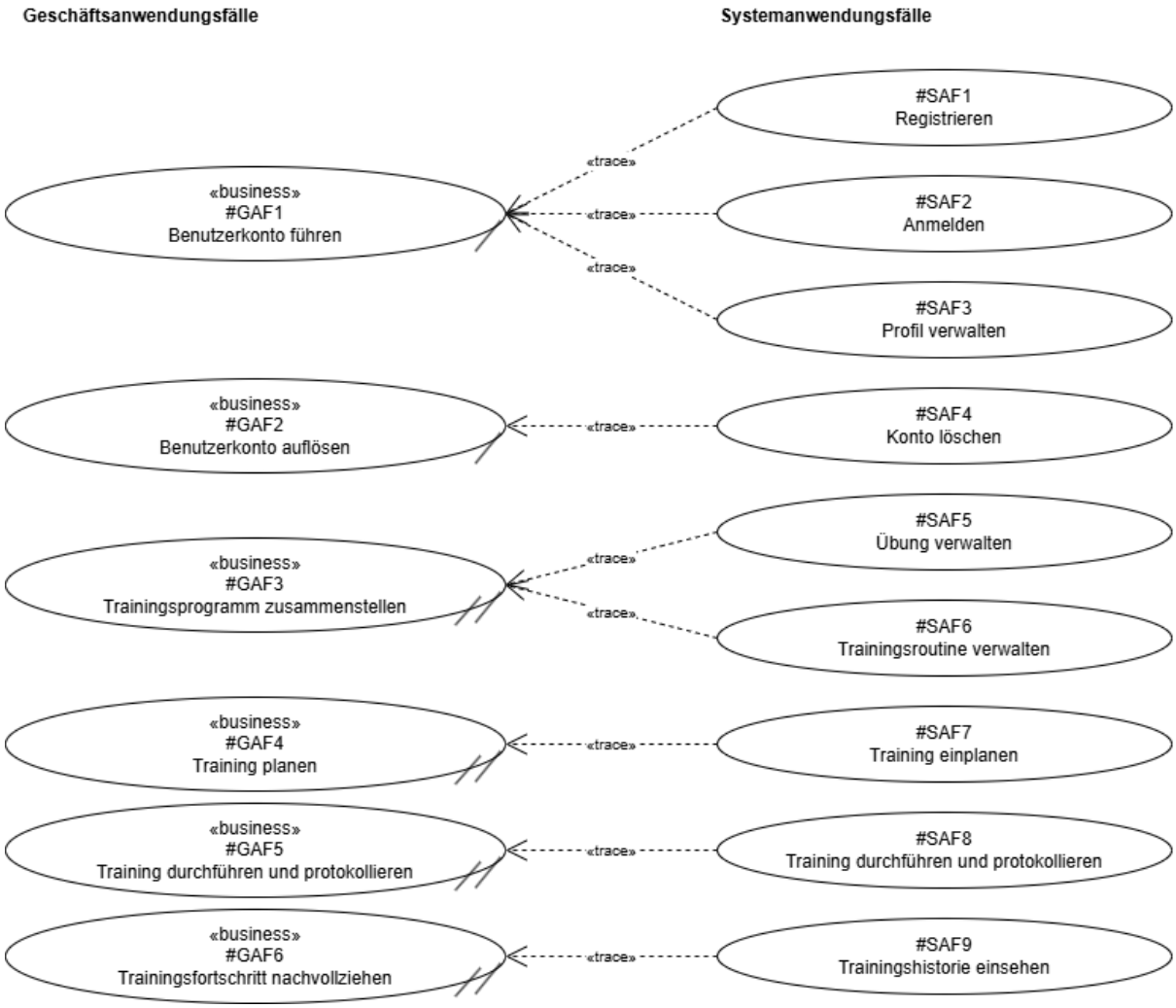


Abbildung 8: Zerlegung der GAF in SAF

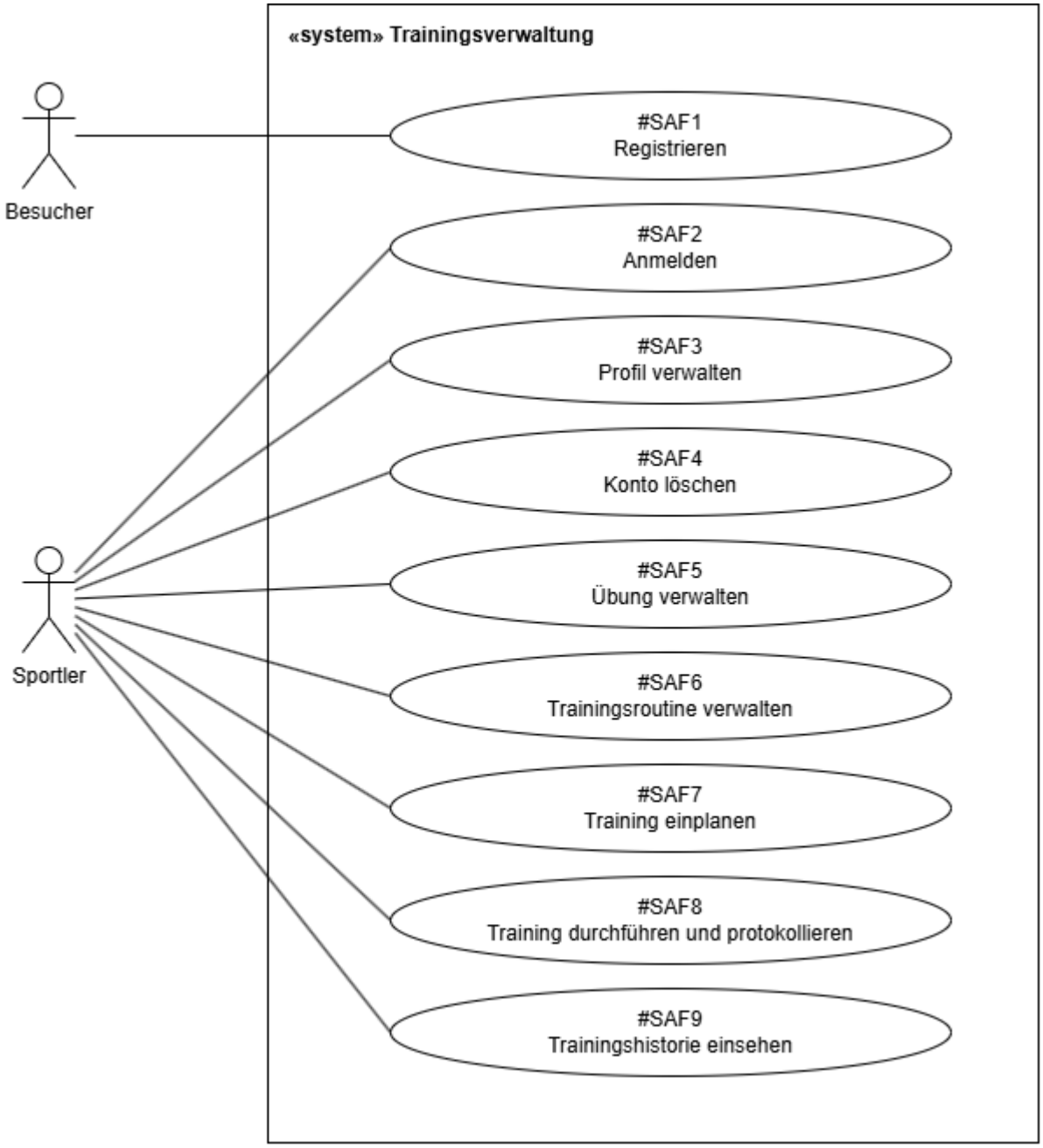


Abbildung 9: Systemanwendungsfälle

5.2.1 SAF1 Registrieren

Tabelle 42: SAF1 Registrieren

Feld	**Inhalt**
Kurzbeschreibung	Ein Besucher eröffnet ein neues Konto.
Akteure	Besucher
Auslöser	Der Besucher will die Plattform für seine Trainings nutzen.
Vorbedingungen	Der Besucher hat noch kein Konto.

Eingehende Informationen	E-Mail-Adresse, Passwort
Ergebnisse	Bestätigung der Registrierung, Sportler ist direkt angemeldet
Nachbedingungen	Konto existiert, Passwort nur als Hash gespeichert
Ablauf	1. Registrierungsdaten eingeben. 2. Registrierungsdaten prüfen. 3. Konto eröffnen. 4. Auf der Plattform angemeldet
Ausnahmen	E-Mail ungültig oder bereits vergeben, Passwort verletzt eine Richtlinie
Anforderungen	#FA-1.1, #FA-1.2, #FA-1.3, #FA-1.4, #NFA-1.1
Risiko	Gering, Standardfunktion von Spring Security
Wichtigkeit	Unverzichtbar (M1)
Aufwand	Mittel
Stabilität	Stabil

5.2.2 SAF2 Anmelden

Tabelle 43: SAF2 Anmelden

Feld	Inhalt
Kurzbeschreibung	Ein bereits registrierter Sportler meldet sich erfolgreich mit seinen bestehenden Anmeldedaten an und erhält 8 Stunden Zugriff auf seine Trainingsdaten.
Akteure	Sportler
Auslöser	Der Sportler startet bald ein Training und möchte die Plattform dafür öffnen, um seine Routine einzusehen.
Vorbedingungen	Konto existiert
Eingehende Informationen	E-Mail-Adresse, Passwort
Ergebnisse	Angemeldet auf der Plattform, Einsicht auf persönliche Trainingsdaten.
Nachbedingungen	Sitzung für 8 Stunden gültig (Token im HttpOnly-Cookie), Fehlversuchszähler zurückgesetzt
Ablauf	1. Anmeldedaten eingeben 2. Anmelden 3. Persönlichen Trainingsdaten anzeigen
Ausnahmen	Daten falsch eingegeben → generische Meldung

	5. Fehlversuch in Folge → Konto 15 Minuten gesperrt
Anforderungen	#FA-1.5, #FA-1.6, #NFA-1.2, #NFA-1.3, #NFA-1.4, #NFA-1.7
Risiko	Mittel, Security-Konzept korrekt umsetzen
Wichtigkeit	Unverzichtbar (M1)
Aufwand	Mittel
Stabilität	Stabil

5.2.3 SAF3 Profil verwalten

Tabelle 44: SAF3 Profil verwalten

Feld	Inhalt
Kurzbeschreibung	Der Sportler hat auf der Plattform die Möglichkeit, seine Kontodaten einzusehen. Dort kann er seine E-Mail-Adresse, persönliche Daten oder Passwort ändern.
Akteure	Sportler
Auslöser	Die E-Mail-Adresse hat sich geändert oder der Sportler will sein Passwort wechseln.
Vorbedingungen	Sportler ist angemeldet
Eingehende Informationen	Neue E-Mail-Adresse, persönliche Daten oder Passwort
Ergebnisse	Bestätigung der Änderung
Nachbedingungen	Kontodaten aktualisiert
Ablauf	1. Profil anzeigen 2. Änderung erfassen 3. Änderung prüfen 4. Kontodaten speichern
Ausnahmen	Neue E-Mail bereits vergeben, neues Passwort verletzt eine Richtlinie
Anforderungen	#FA-1.2, #FA-1.3, #FA-1.7, #FA-1.8, #FA-1.9
Risiko	Gering
Wichtigkeit	Wichtig (M1)
Aufwand	Gering
Stabilität	Stabil

5.2.4 SAF4 Konto löschen

Tabelle 45: SAF4 Konto löschen

Feld	Inhalt
Kurzbeschreibung	Der Sportler will sein Konto löschen. Das System entfernt den Benutzer und alle seine Trainingsdaten endgültig wie im Löschkonzept vorgesehen.
Akteure	Sportler
Auslöser	Der Sportler will die Plattform nicht mehr nutzen und seine Daten entfernen.
Vorbedingungen	Sportler ist angemeldet
Eingehende Informationen	Löschwunsch, Passwort zur Bestätigung
Ergebnisse	Bestätigung der Löschung, Sportler ist abgemeldet
Nachbedingungen	Konto und alle zugehörigen Daten hart gelöscht, ein noch gültiges Token ist wirkungslos
Ablauf	1. Löschung anfordern 2. Hinweis auf Endgültigkeit anzeigen 3. Mit Passwort bestätigen 4. Passwort prüfen 5. Konto mit allen Daten löschen 6. Sportler abmelden und Löschung bestätigen
Ausnahmen	Passwort falsch → keine Löschung Fehler während der Löschung → nichts gelöscht (Rollback)
Anforderungen	#FA-1.10, #NFA-1.5, #NFA-1.6
Risiko	Mittel, Vollständigkeit über alle Fremdschlüsselketten
Wichtigkeit	Unverzichtbar (M1, Löschkonzept)
Aufwand	Mittel
Stabilität	Stabil

5.2.5 SAF5 Übung verwalten

Tabelle 46: SAF5 Übung verwalten

Feld	Inhalt
Kurzbeschreibung	Der Sportler hat einen Übungskatalog, in dem er alle seine Übungen erfassen, ändern und löschen kann. Pro Übung kann man den Namen, die Muskelgruppe und eine Beschreibung erfassen.
Akteure	Sportler

Auslöser	Der Sportler will eine neue Übung trainieren oder eine bestehende anpassen oder entfernen.
Vorbedingungen	Sportler ist angemeldet
Eingehende Informationen	Name, Muskelgruppe, Beschreibung
Ergebnisse	Aktueller Übungskatalog
Nachbedingungen	Übung gespeichert, geändert, gelöscht oder soft gelöscht
Ablauf	1. Übungskatalog anzeigen 2. Übung wählen oder neue anlegen 3. Angaben erfassen oder ändern 4. Angaben prüfen 5. Übung speichern. 6. Variante Löschen: Löschen anfordern, bestätigen, Übung löschen
Ausnahmen	Name bereits vorhanden, Pflichtfeld leer Übung in einer Routine enthalten → Löschen nicht möglich
Anforderungen	#FA-2.1 bis #FA-2.6
Risiko	Gering
Wichtigkeit	Unverzichtbar (M2, M5)
Aufwand	Gering
Stabilität	Stabil

5.2.6 SAF6 Trainingsroutine verwalten

Tabelle 47: SAF6 Trainingsroutine verwalten

Feld	Inhalt
Kurzbeschreibung	Der Sportler stellt aus Übungen seines Katalogs Trainingsroutinen in fester Reihenfolge zusammen, ändert oder löscht sie.
Akteure	Sportler
Auslöser	Der Sportler will ein neues Training zusammenstellen oder ein bestehendes anpassen.
Vorbedingungen	Sportler ist angemeldet, mindestens eine Übung im Katalog
Eingehende Informationen	Name der Routine, gewählte Übungen, Reihenfolge
Ergebnisse	Einsatzbereite Routine

Nachbedingungen	Routine mit geordneter Übungsliste gespeichert, geändert, gelöscht oder soft gelöscht. Routine kann für eine Trainingseinheit eingeplant werden.
Ablauf	1. Routinen anzeigen 2. Routine wählen oder neu anlegen 3. Name erfassen 4. Übungen aus dem Katalog hinzufügen oder entfernen 5. Reihenfolge festlegen 6. Angaben prüfen 7. Routine speichern. 8. Variante Löschen: Löschen anfordern, bestätigen, Routine löschen
Ausnahmen	Routine ohne Übung, Name bereits vorhanden Routine mit geplanten Trainings löschen → Rückfrage (Es werden die bestehenden Trainingseinheiten mit Status «geplant» gelöscht)
Anforderungen	#FA-3.1 bis #FA-3.7
Risiko	Mittel, Änderungen dürfen die Historie nicht verfälschen
Wichtigkeit	Unverzichtbar (M2)
Aufwand	Mittel
Stabilität	Stabil

5.2.7 SAF7 Training einplanen

Tabelle 48: SAF7 Training einplanen

Feld	Inhalt
Kurzbeschreibung	Der Sportler plant eine Routine für ein zukünftiges Datum ein und kann geplante Trainings ändern oder entfernen.
Akteure	Sportler
Auslöser	Der Sportler plant seine nächsten Trainingstage.
Vorbedingungen	Sportler ist angemeldet, mindestens eine Routine vorhanden
Eingehende Informationen	Routine, Datum
Ergebnisse	Übersicht der geplanten Trainings mit dem neuen Eintrag
Nachbedingungen	Trainingseinheit im Status «geplant»
Ablauf	1. Geplante Trainings anzeigen 2. Routine wählen 3. Datum wählen 4. Angaben prüfen 5. Training einplanen.

	6. Variante Löschen: Löschen anfordern, bestätigen, Trainingseinheit löschen
Ausnahmen	Datum in der Vergangenheit -> kann nur gelöscht werden, nicht verändert.
Anforderungen	#FA-4.1 bis #FA-4.5, #NFA-4.1
Risiko	Gering
Wichtigkeit	Unverzichtbar (M3)
Aufwand	Gering
Stabilität	Stabil

5.2.8 SAF8 Training durchführen und protokollieren

Tabelle 49: SAF8 Training durchführen und protokollieren

Feld	Inhalt
Kurzbeschreibung	Der Sportler startet ein geplantes Training oder direkt eine Routine, erfasst pro Übung laufend die Sätze mit Gewicht, Wiederholungen und Pausenzeit und schliesst die Einheit ab.
Akteure	Sportler
Auslöser	Der Sportler beginnt im Studio sein Training.
Vorbedingungen	Sportler ist angemeldet, Routine vorhanden, keine andere Einheit aktiv
Eingehende Informationen	Geplantes Training oder Routine, pro Satz Gewicht (kg), Wiederholungen, Pausenzeit (s), Notiz zur Einheit
Ergebnisse	Abgeschlossene, vollständig protokollierte Trainingseinheit
Nachbedingungen	Einheit im Status «abgeschlossen», alle Sätze gespeichert, Einheit in der Historie sichtbar
Ablauf	1. Training starten 2. Übung mit den Werten der letzten Einheit anzeigen 3. Satz erfassen 4. Satz speichern 5. (2) bis (4) je Satz und Übung wiederholen 6. Training abschliessen.
Ausnahmen	Unplausible Werte, Abschluss ohne Satz Speichern fehlgeschlagen → Eingabe bleibt erhalten Sitzung nach 8 Stunden abgelaufen → neu anmelden, Einheit bleibt aktiv
Anforderungen	#FA-3.5, #FA-5.1 bis #FA-5.9, #FA-6.5, #NFA-5.1 bis #NFA-5.5

Risiko	Hoch, flüssige Bedienung im Studio (T9), Frontend-Aufwand (R01), Mobile-Darstellung (R04)
Wichtigkeit	Unverzichtbar (M5, Kern der Projektvision)
Aufwand	Hoch
Stabilität	Instabil, Bedienung wird mit dem Konzept des Frontends konkretisiert

5.2.9 SAF9 Trainingshistorie einsehen

Tabelle 50: SAF9 Trainingshistorie einsehen

Feld	Inhalt
Kurzbeschreibung	Der Sportler sieht zu einer Routine alle abgeschlossenen Einheiten und öffnet eine Einheit mit allen Sätzen im Detail.
Akteure	Sportler
Auslöser	Der Sportler will wissen, was er bei einer Routine zuletzt geleistet hat.
Vorbedingungen	Sportler ist angemeldet
Eingehende Informationen	Gewählte Routine, gewählte Einheit Man kann auch soft-gelöschte Routinen mit mindestens einer Trainingseinheit auswählen.
Ergebnisse	Liste der abgeschlossenen Einheiten (neueste zuerst), Detailansicht mit Übungen, Sätzen und Notiz
Nachbedingungen	Keine Änderung (nur lesend) Variante Löschen: Einheit als gelöscht markiert und wird nicht mehr angezeigt.
Ablauf	1. Routine wählen 2. Abgeschlossene Einheiten anzeigen 3. Einheit wählen 4. Details anzeigen. 5. Variante: Einheit aus der Historie löschen (mit Bestätigung)
Ausnahmen	Noch keine abgeschlossene Einheit
Anforderungen	#FA-3.5, #FA-6.1 bis #FA-6.5, #NFA-6.1
Risiko	Gering
Wichtigkeit	Unverzichtbar (M4)
Aufwand	Gering
Stabilität	Stabil

5.3 Anforderungen

Die insgesamt 80 Anforderungen sind in sieben Features gebündelt. Ein Feature ist die Sammlung fachlich zusammenhängender Anforderungen. Die Anforderungen sind aufgeteilt in FA (Funktionale Anforderungen) und NFA (Nicht-funktionale Anforderungen). Alle Anforderungen sind verbindlich und müssen eingehalten werden, da sie ausschliesslich aus den Muss-Zielen abgeleitet wurden.

Die Spalte «Prüfung» zeigt auf, gemäss welcher Teststrategie diese Anforderung getestet wird. U = Unit-Test, S = manueller Systemtest, SM = Smoke-Test und R = Review. Somit ist jede Anforderung testbar.

Die nicht-funktionalen Anforderungen erhalten die Spalte Kategorie, wo sie in Zuverlässigkeit, Benutzbarkeit, Effizienz, Sicherheit, Übertragbarkeit, Änderbarkeit und Randbedingung aufgeteilt werden.

Tabelle 51: Anforderungen pro Feature

ID	FA	NFA
#FT1	10	7
#FT2	6	–
#FT3	7	–
#FT4	5	1
#FT5	9	5
#FT6	5	1
#FT7	–	24
Total	42	38

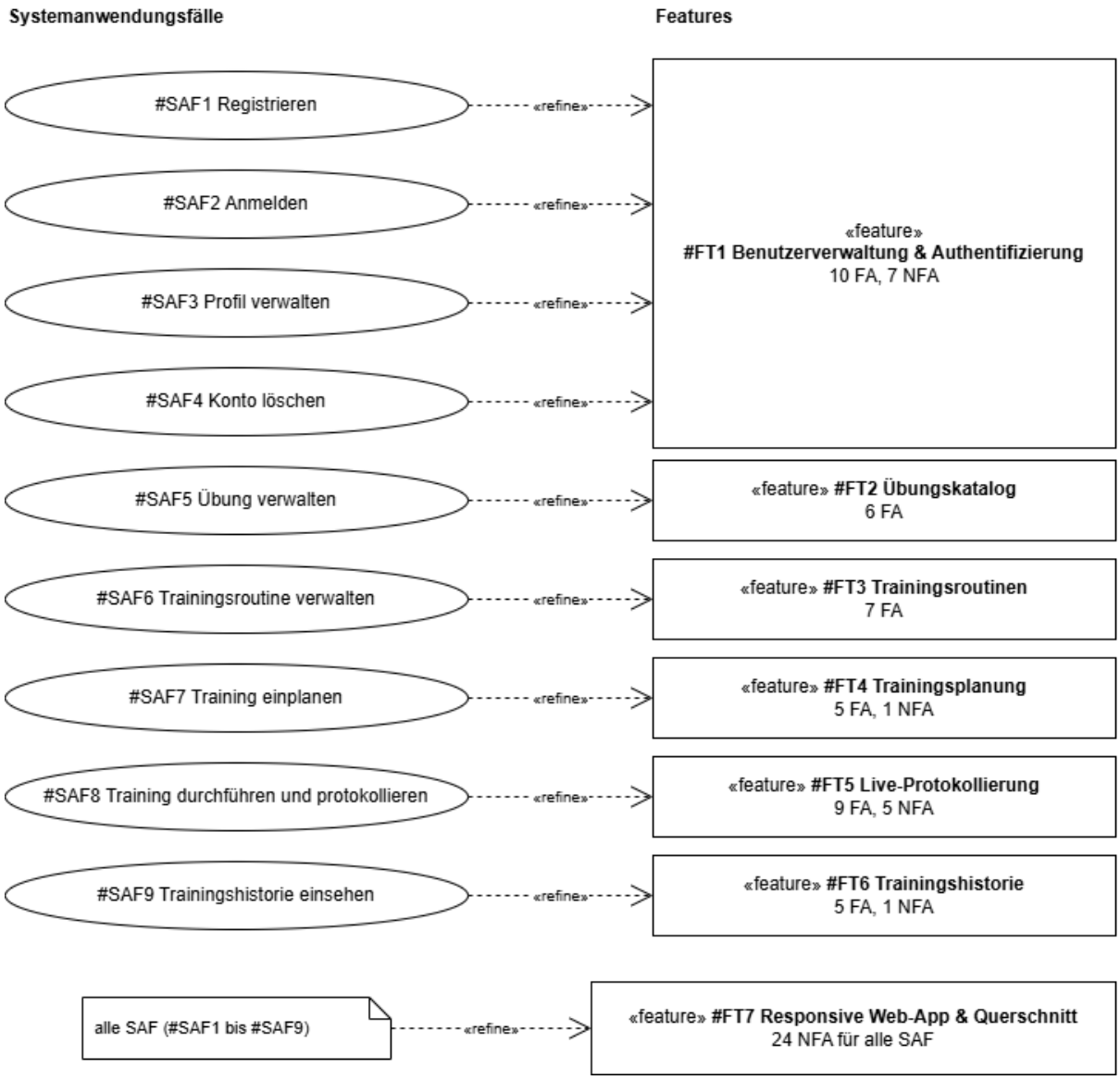


Abbildung 10: Zuordnung der SAF zu den Features

5.3.1 FT1 Benutzerverwaltung & Authentifizierung

Tabelle 52: FT1 Benutzerverwaltung & Authentifizierung

Feld	Inhalt
Art	Feature
Beschreibung	Sportler registrieren sich, melden sich sicher an, pflegen ihre Kontodaten und können ihr Konto mit allen Daten endgültig löschen. Grundlage für den Schutz aller Trainingsdaten.
Enthaltene Anforderungen	#FA-1.1 bis #FA-1.10, #NFA-1.1 bis #NFA-1.7
Verfeinert durch	#SAF1, #SAF2, #SAF3, #SAF4

Funktionale Anforderungen

Tabelle 53: FA zu FT1

ID	Name	Beschreibung	Prüfung
#FA-1.1	Registrieren	Ein Besucher muss mit E-Mail-Adresse und Passwort ein Konto eröffnen können. Weitere Personendaten sind nicht Pflicht, können aber angegeben werden.	U, S
#FA-1.2	Passwortrichtlinie	15 bis 64 Zeichen Keine Vorgaben zur Zusammensetzung Kein periodischer Wechsel Einfügen aus der Zwischenablage erlaubt Passwort kann zur Kontrolle angezeigt werden.	U, S
#FA-1.3	Eindeutige E-Mail-Adresse	Gültiges Format Pro E-Mail-Adresse höchstens ein Konto Gross- und Kleinschreibung wird nicht unterschieden.	U, S
#FA-1.4	Datenschutzerklärung einsehen	Die Datenschutzerklärung muss ohne Anmeldung und vor der Registrierung aufrufbar sein. Darin enthalten sind Informationen zum Verantwortlichen, Zweck, Daten, Speicherort und Aufbewahrung.	S
#FA-1.5	Anmelden	Ein registrierter Sportler muss sich mit E-Mail-Adresse und Passwort anmelden können.	U, S
#FA-1.6	Abmelden	Ein angemeldeter Sportler muss sich jederzeit abmelden können. Danach ist ohne neue Anmeldung kein Zugriff möglich.	S
#FA-1.7	Profil einsehen	Der Sportler muss seine hinterlegten persönliche Daten einsehen können. Diese sind Vorname und Nachname.	S
#FA-1.8	Daten ändern	Der Sportler muss seine persönlichen Daten ändern können. Diese kann er direkt speichern. Wenn er seine E-Mail-Adresse ändert, muss sie weiterhin eindeutig sein und er muss zur Bestätigung sein Passwort eingeben.	U, S
#FA-1.9	Passwort ändern	Der Sportler muss sein Passwort ändern können und gibt dazu das aktuelle Passwort ein. Das Token wird erneuert und ist wieder 8 Stunden gültig.	U, S

#FA-1.10	Konto löschen	Der Sportler muss sein Konto jederzeit selbst löschen können und bestätigt dies mit seinem Passwort. Es werden alle Daten des Sportlers in einer Transaktion hart gelöscht.	U, S

Nicht-funktionale Anforderungen

Tabelle 54: NFA zu FT1

ID	Name	Beschreibung	Kategorie	Prüfung
#NFA-1.1	Passwortspeicherung	Passwörter müssen mit Argon2id gehasht gespeichert werden, nie im Klartext, nie in Logs.	Sicherheit	U
#NFA-1.2	Sitzung	Der angemeldete Sportler muss sich per JWT in einem HttpOnly-, Secure- und SameSite=Strict-Cookie ausweisen Das Token ist 8 Stunden gültig.	Sicherheit	U, S
#NFA-1.3	Schutz vor Brute-Force	Nach 5 aufeinanderfolgenden Fehlversuchen muss das Konto für 15 Minuten gesperrt werden. Dabei wird die E-Mail-Adresse gesperrt, egal ob es sie gibt oder nicht.	Sicherheit	U, S
#NFA-1.4	Generische Fehlermeldung	Eine fehlgeschlagene Anmeldung darf nicht verraten, ob die E-Mail-Adresse existiert.	Sicherheit	S
#NFA-1.5	Sofortige Wirkung der Löschung	Nach einer Kontolöschung muss jede Anfrage mit dem alten Token abgewiesen werden.	Sicherheit	U, S
#NFA-1.6	Atomare Löschung	Die Kontolöschung muss in einer einzigen Transaktion erfolgen. Entweder sind alle Daten gelöscht oder keine.	Zuverlässigkeit	U
#NFA-1.7	Sicherheitsereignisse protokollieren	Fehlgeschlagene Anmeldungen und verweigerte Zugriffe müssen geloggt	Sicherheit	S

		werden, ohne Passwörter oder Tokens.		

5.3.2 FT2 Übungskatalog

Tabelle 55: FT2 Übungskatalog

Feld	Inhalt
Art	Feature
Beschreibung	Der persönliche Übungskatalog ist der Baustein aller Routinen und Protokolle. Jede Übung hat Name, Muskelgruppe und Beschreibung.
Enthaltene Anforderungen	#FA-2.1 bis #FA-2.6
Verfeinert durch	#SAF5

Funktionale Anforderungen

Tabelle 56: FA zu FT2

ID	Name	Beschreibung	Prüfung
#FA-2.1	Übung erfassen	Der Sportler muss eine Übung mit Namen (Pflicht), Muskelgruppe (Pflicht) und Beschreibung (optional) erfassen können.	U, S
#FA-2.2	Muskelgruppen	Jede Übung hat genau eine Hauptmuskelgruppe aus einer festen Liste: • Brust • Rücken • Schultern • Bizeps • Trizeps • Bauch • Beine • Gesäss	S
#FA-2.3	Übungskatalog anzeigen	Das System muss alle nicht soft-gelöschten Übungen des Sportlers mit Muskelgruppe anzeigen, alphabetisch sortiert.	S
#FA-2.4	Übung bearbeiten	Der Sportler muss Name, Muskelgruppe und Beschreibung ändern können. Die Änderung gilt auch in bestehenden Routinen und Historien, da es dieselbe Übung ist.	U, S
#FA-2.5	Eindeutiger Namen	Der Übungsname ist je Sportler eindeutig, ohne Unterscheidung von Gross und Kleinschreibung. Soft-gelöschte Datensätze werden nicht berücksichtigt bei dieser Prüfung.	U

#FA-2.6	Übung löschen	Der Sportler muss eine Übung soft-löschen können Eine Übung, die in einer Routine enthalten ist, kann erst soft-gelöscht werden, wenn sie aus allen nicht soft-gelöschten Routinen entfernt ist. Kommt die Übung nur in vergangenen Trainingseinheiten vor darf sie aus dem Übungskatalog soft-gelöscht werden. Die Historie bleibt vollständig.	U, S

5.3.3 FT3 Trainingsroutinen

Tabelle 57: FT3 Trainingsroutinen

Feld	Inhalt
Art	Feature
Beschreibung	Routinen fassen Übungen in fester Reihenfolge zu einem wiederverwendbaren Training zusammen. Sie sind Grundlage für Planung, Live-Protokoll und Historie.
Enthaltene Anforderungen	#FA-3.1 bis #FA-3.7
Verfeinert durch	#SAF6

Funktionale Anforderungen

Tabelle 58: FA zu FT3

ID	Name	Beschreibung	Prüfung
#FA-3.1	Routine erstellen	Der Sportler muss eine Routine mit Namen und mindestens einer Übung aus seinem Katalog erstellen können. Jede Übung darf höchstens einmal vorkommen. Die Reihenfolge ist eindeutig.	U, S
#FA-3.2	Routinen anzeigen	Das System muss alle nicht soft-gelöschten Routinen des Sportlers mit ihren Übungen anzeigen.	S
#FA-3.3	Routine bearbeiten	Der Sportler muss Name und Übungen einer Routine ändern können, bereits protokollierte Einheiten bleiben unverändert.	U, S
#FA-3.4	Eindeutiger Namen	Der Routinen Name ist je Sportler eindeutig, ohne Unterscheidung von Gross und Kleinschreibung. Soft-gelöschte Datensätze werden nicht berücksichtigt bei dieser Prüfung.	U
#FA-3.5	Historische Konsistenz	Eine protokollierte Trainingseinheit bleibt unverändert. Nur wenn die Übung im Übungskatalog bearbeitet wird, verändert das auch die Historie.	U, S

#FA-3.6	Reihenfolge festlegen	Der Sportler muss die Reihenfolge der Übungen festlegen und ändern können, z.B. mit «nach oben» / «nach unten»	S
#FA-3.7	Routine löschen	Der Sportler muss eine Routine löschen können. Wurde die Routine schon für eine Trainingseinheit verwendet, wird sie nur aus den bestehenden Routinen entfernt, sodass die Historie vollständig bleibt. Die Trainingseinheiten mit Status «geplant» dieser Routine werden entfernt (mit Rückfrage)	U, S

5.3.4 FT4 Trainingsplanung

Tabelle 59: FT4 Trainingsplanung

Feld	Inhalt
Art	Feature
Beschreibung	Der Sportler legt fest, an welchem zukünftigen Tag er welche Routine trainiert. Brücke zwischen Routine und Durchführung.
Enthaltene Anforderungen	#FA-4.1 bis #FA-4.5, #NFA-4.1
Verfeinert durch	#SAF7

Funktionale Anforderungen

Tabelle 60: FA zu FT4

ID	Name	Beschreibung	Prüfung
#FA-4.1	Training einplanen	Der Sportler muss eine Routine für ein Datum ab heute einplanen können; die Einheit erhält den Status «geplant»	U, S
#FA-4.2	Planungsdatum	Ein Training kann nur für heute oder später eingeplant werden. Mehrere Trainings pro Tag sind erlaubt.	U, S
#FA-4.3	Geplante Trainings anzeigen	Das System muss alle geplanten Trainings chronologisch anzeigen, das nächste zuerst, je mit Datum und Routine.	S
#FA-4.4	Geplantes Training ändern	Der Sportler muss Datum oder Routine eines geplanten Trainings ändern können, sofern es nicht in der Vergangenheit liegt.	U, S
#FA-4.5	Geplantes Training entfernen	Der Sportler muss ein geplantes Training entfernen können, solange es nicht gestartet ist (Status «geplant»).	U, S

Nicht-funktionale Anforderungen

Tabelle 61: NFA zu FT4

ID	Name	Beschreibung	Kategorie	Prüfung
#NFA-4.1	Nächstes Training auf einen Blick	Nach der Anmeldung muss das nächste geplante Training ohne weitere Navigation sichtbar und mit einem Tipp startbar sein. Geplante Trainings der Vergangenheit sind hier ebenfalls sichtbar und werden ganz oben angezeigt.	Benutzbarkeit	S

5.3.5 FT5 Live-Protokollierung

Tabelle 62: FT5 Live-Protokollierung

Feld	Inhalt
Art	Feature
Beschreibung	Kernfunktion im Studio: Der Sportler erfasst während des Trainings Satz für Satz Gewicht, Wiederholungen und Pausenzeit. Jede Eingabe wird sofort gespeichert.
Enthaltene Anforderungen	#FA-5.1 bis #FA-5.9, #NFA-5.1 bis #NFA-5.5
Verfeinert durch	#SAF8

Funktionale Anforderungen

Tabelle 63: FA zu FT5

ID	Name	Beschreibung	Prüfung
#FA-5.1	Training starten	Der Sportler muss ein geplantes Training oder direkt eine Routine starten können, die Einheit wird «aktiv» und zeigt die Übungen in der Reihenfolge der Routine. Pro Sportler ist höchstens eine Einheit gleichzeitig aktiv. Wenn ein Training noch nicht abgeschlossen wurde, wird der Start blockiert.	U, S
#FA-5.2	Satz erfassen	Der Sportler muss pro Übung beliebig viele Sätze mit Gewicht in kg, Wiederholungen und Pausenzeit in Sekunden erfassen können, die Satznummer vergibt das System fortlaufend.	U, S
#FA-5.3	Plausible Satzwerte	Gewicht 0 kg bis 1000 kg mit höchstens 2 Nachkommastellen Wiederholungen ganze Zahl 1 bis 100 Pausenzeit optional, 0 bis 3600 s	U, S

#FA-5.4	Satz korrigieren	Der Sportler muss erfasste Sätze während der aktiven Einheit ändern, löschen und verschieben können.	U, S
#FA-5.5	Letzte Werte anzeigen	Im Frontend werden alle Sätze vorbelegt mit den Werten derselben Übung aus der letzten abgeschlossenen Einheit mit derselben Routine. Diese Werte werden aber erst persistiert, wenn man im Frontend speichern drückt. Fehlt dieser Wert, bleiben die Sätze leer und man muss selbst erfassen. Soft-gelöschte Einheiten werden nicht berücksichtigt.	U, S
#FA-5.6	Notiz erfassen	Der Sportler muss zur Einheit eine optionale Notiz als Freitext erfassen können.	S
#FA-5.7	Training abschliessen	Der Sportler muss die Einheit abschliessen können; danach ist sie «abgeschlossen» und in der Historie sichtbar. Eine Einheit kann nur mit mindestens einem erfassten Satz abgeschlossen werden. Übungen ohne Satz gelten als ausgelassen und werden nicht gespeichert.	U, S
#FA-5.8	Training abbrechen	Der Sportler muss eine aktive Einheit nach Bestätigung abbrechen können. Erfasste Sätze werden verworfen. Der Status wird auf «geplant» gesetzt. Wenn es sich um ein spontanes Training handelt, wird beim Abbruch nichts persistiert.	U, S
#FA-5.9	Aktives Training fortsetzen	Nach Neuladen, Gerätewechsel oder neuer Anmeldung muss der Sportler eine aktive Einheit mit allen gespeicherten Sätzen fortsetzen können.	S

Nicht-funktionale Anforderungen

Tabelle 64: NFA zu FT5

ID	Name	Beschreibung	Kategorie	Prüfung
#NFA-5.1	Sofortige Speicherung	Jeder Satz muss beim Bestätigen sofort serverseitig gespeichert werden. Gespeicherte Sätze gehen bei Verbindungsabbruch, Neuladen oder Ablauf der Sitzung nicht verloren.	Zuverlässigkeit	S
#NFA-5.2	Fehler beim Speichern	Schlägt das Speichern fehl, muss eine verständliche Meldung	Zuverlässigkeit	S

		erscheinen und die Eingabe zum erneuten Senden erhalten bleiben.		
#NFA-5.3	Satz mit einem Tipp	Ein Satz mit unveränderten, vorbelegten Werten muss mit einem einzigen Tipp gespeichert werden können.	Benutzbarkeit	S
#NFA-5.4	Zahleneingabe	Felder für Gewicht, Wiederholungen und Pausenzeit müssen auf dem Smartphone die numerische Tastatur öffnen.	Benutzbarkeit	S
#NFA-5.5	Antwortzeit	Vom Bestätigen eines Satzes bis zur Rückmeldung darf höchstens 1 Sekunde vergehen.	Effizienz	S

5.3.6 FT6 Trainingshistorie

Tabelle 65: FT6 Trainingshistorie

Feld	Inhalt
Art	Feature
Beschreibung	Zeigt pro Routine, was in früheren Einheiten geleistet wurde als Grundlage für die nächste Steigerung. Nur Einsicht, keine grafische Auswertung.
Enthaltene Anforderungen	#FA-6.1 bis #FA-6.5, #NFA-6.1
Verfeinert durch	#SAF9

Funktionale Anforderungen

Tabelle 66: FA zu FT6

ID	Name	Beschreibung	Prüfung
#FA-6.1	Historie einer Routine anzeigen	Das System muss zu einer gewählten Routine alle abgeschlossenen Einheiten anzeigen, neueste zuerst, je mit tatsächlichem Startdatum.	S
#FA-6.2	Einheit im Detail anzeigen	Das System muss zu einer Einheit Datum, Routine, alle Übungen in Reihenfolge mit ihren Sätzen (Gewicht, Wiederholungen, Pausenzeit) und die Notiz anzeigen.	S
#FA-6.3	Historie soft-gelöschter Routinen	Die Historie muss auch für soft-gelöschte Routinen einsehbar bleiben. Diese werden zwar markiert, aber können ganz normal ausgewählt werden für die Einsicht.	U, S

#FA-6.4	Einheit löschen	Der Sportler muss eine abgeschlossene Einheit nach Bestätigung aus der Historie soft-löschen können.	U, S
#FA-6.5	Schreibschutz	Eine abgeschlossene Einheit kann nicht mehr bearbeitet werden, nur noch gelöscht werden.	U, S

Nicht-funktionale Anforderungen

Tabelle 67: NFA zu FT6

ID	Name	Beschreibung	Kategorie	Prüfung
#NFA-6.1	Ladezeit Historie	Die Historie einer Routine mit 300 Einheiten muss in höchstens 2 Sekunden angezeigt sein, manuell gemessen mit Testdaten.	Effizienz	S

5.3.7 FT7 Responsive Web-App & Querschnitt

Tabelle 68: FT7 Responsive Web-App & Querschnitt

Feld	Inhalt
Art	Feature
Beschreibung	Gilt für alle SAF: Bedienung auf Smartphone und Desktop (M6) sowie alle Technologie- und Betriebsvorgaben.
Enthaltene Anforderungen	#NFA-7.1 bis #NFA-7.24
Verfeinert durch	alle SAF (#SAF1 bis #SAF9)

Nicht-funktionale Anforderungen

Tabelle 69: NFA zu FT7

ID	Name	Beschreibung	Kategorie	Prüfung
#NFA-7.1	Responsive Layout	Alle Funktionen müssen von 360 bis 1920 Pixel Bildschirmbreite ohne horizontales Scrollen bedienbar sein	Übertragbarkeit	S
#NFA-7.2	Browser	Die Applikation muss in den aktuellen Versionen von Firefox und Chrome auf einem Windows-Desktop sowie auf Android funktionieren.	Übertragbarkeit	S
#NFA-7.3	Touch-Ziele	Bedienelemente müssen mindestens 44 × 44 Pixel gross sein.	Benutzbarkeit	S

#NFA-7.4	Rückmeldungen	Jede speichernde oder löschende Aktion muss den Erfolg bestätigen oder eine verständliche Fehlermeldung ohne technische Details zeigen.	Benutzbarkeit	S
#NFA-7.5	Bestätigung vor dem Löschen	Jede Löschaktion muss vor der Ausführung bestätigt werden.	Benutzbarkeit	S
#NFA-7.6	Datentrennung	Jeder Datenzugriff muss den Besitz anhand der Benutzer-ID aus dem Token prüfen Fremde oder nicht vorhandene Datensätze ergeben «nicht gefunden» (404).	Sicherheit	U, S
#NFA-7.7	Zugriff nur mit Anmeldung	Ausser Registrierung, Anmeldung, Datenschutzerklärung und Health-Check müssen alle Funktionen eine gültige Anmeldung verlangen (Deny by default).	Sicherheit	U, S
#NFA-7.8	Verschlüsselte Übertragung	Die Kommunikation muss ausschliesslich über HTTPS laufen; Aufrufe über HTTP werden auf HTTPS umgeleitet.	Sicherheit	SM
#NFA-7.9	Serverseitige Prüfung	Alle Eingaben und Geschäftsregeln müssen im Backend verbindlich geprüft werden Die Prüfung im Frontend dient nur der Benutzerführung. Das Frontend enthält keine Geschäftslogik.	Sicherheit	U
#NFA-7.10	Keine Drittanbieter	Im Browser dürfen keine Dienste von Drittanbietern geladen werden: Kein Tracking, keine Analyse, keine Werbung.	Sicherheit	S
#NFA-7.11	Sprache und Formate	Die Oberfläche muss deutsch sein Datum im Format TT.MM.JJJJ, Gewichte in kg.	Benutzbarkeit	S

#NFA-7.12	Log-Aufbewahrung	Die Logfiles sind höchstens 14 Tage lang aufbewahrt.	Sicherheit	R
#NFA-7.13	Technologie-Stack	Backend mit Spring Boot 4 und Java 25, Frontend mit React 19, TypeScript (strict) und Material UI, Datenbank PostgreSQL 18. Alle Versionen sind fixiert. Aktualisiert wird nur bei kritischen Sicherheitslücken.	Randbedingung	R
#NFA-7.14	Datenhaltung	Jede fachliche Entität hat einen eigenen Primärschlüssel. Beziehungen werden über Fremdschlüssel in der Datenbank erzwungen. Schemaänderungen erfolgen nur über versionierte Migrationen (Flyway). Datenbankzugriffe sind ausschliesslich parametrisiert (Spring Data JPA).	Änderbarkeit	R
#NFA-7.15	Schichtenarchitektur	Das Backend muss die Schichten Controller, Service, Persistenz ohne Überspringen einhalten und nach fachlichen Bereichen paketiert sein. Zugriffe auf einen anderen Bereich erfolgen nur über dessen Service.	Änderbarkeit	R
#NFA-7.16	API-Vertrag	Die Schnittstelle muss als versionierte REST/JSON-API unter /api umgesetzt sein, nach aussen nur DTOs liefern und als OpenAPI-Spezifikation dokumentiert sein. Die Swagger UI ist in Produktion nicht erreichbar.	Änderbarkeit	R, SM
#NFA-7.17	Fehlerbehandlung	Fehler müssen im Backend behandelt und in einem einheitlichen Format zurückgegeben werden. Stacktraces gelangen nie ins Frontend.	Sicherheit	U, S

		Schlägt ein Vorgang auf der Datenbank fehl, gibt es einen Rollback.		
#NFA-7.18	Betriebsumgebung	Die Applikation muss als Docker-Compose-Setup auf einem einzelnen Server von Infomaniak mit Rechenzentrum in der Schweiz laufen. Frontend und API sind über dieselbe Domain erreichbar.	Randbedingung	R, SM
#NFA-7.19	Netzwerkzugriff	Einziger Einstiegspunkt ist der Reverse Proxy (Caddy). Die Firewall lässt eingehend nur die Ports 22, 80 und 443 zu. Backend und Datenbank haben keine veröffentlichten Ports.	Sicherheit	SM
#NFA-7.20	Secrets	Zugangsdaten und Schlüssel (DB-Passwort, JWT-Schlüssel) dürfen nie im Repository liegen. Sie werden zur Laufzeit über Umgebungsvariablen bereitgestellt.	Sicherheit	R
#NFA-7.21	Reproduzierbares Deployment	Die Applikation muss sich aus dem Repository auf einem leeren Host neu aufbauen lassen. Alle Umgebungen verwenden dasselbe Build-Artefakt, nur die Konfiguration unterscheidet sich.	Übertragbarkeit	R
#NFA-7.22	Automatisierte Qualitätssicherung	Der main-Branch ist geschützt und nur per Pull Request mit grüner Pipeline änderbar. Die Pipeline führt Build, ESLint, Typprüfung und alle Unit-Tests aus. Ein roter Test blockiert Merge und Deployment. Container-Images werden nur in der Pipeline gebaut und nur über sie deployt.	Änderbarkeit	R
#NFA-7.23	Kontrolle nach Deployment	Nach jedem Deployment muss der Health-Endpunkt automatisch über HTTPS geprüft und ein manueller	Zuverlässigkeit	SM

		Smoke-Test (Login, Kernablauf) durchgeführt werden.		
#NFA-7.24	Keine Gesundheitsdaten	Das System darf keine Gesundheitsdaten erfassen, also keine Felder für Körpergewicht, Körpermasse oder Verletzungen.	Sicherheit	R

6 Konzeption

Dieses Kapitel dient als Grundlage für die Entwicklung. Aus den Technologieentscheidungen, Leitplanken und Anforderungen ergeben sich klare Baupläne, die während der Umsetzung alles vorgeben. Aufgebaut wird in der untersten Schicht der Bausteinsicht, der Datenbank. Weiter geht es mit Klassendiagrammen für die Backend-Struktur, aus welcher sich eine eindeutige REST-Schnittstelle ergibt. Mit dieser Grundlage kann schlussendlich das Frontend konzipiert werden.

Die Anforderungen decken nicht alle Detailentscheide ab, welche für ein sauberes technisches Konzept festgelegt werden müssen. Diese werden hier nochmals kurz beschrieben oder klar für die technische Umsetzung definiert.

Kopie oder Verweis: Die Übungsliste der Routine wird beim Start einer Trainingseinheit kopiert, sodass die Historisierung gegeben ist. Geplante Trainingseinheiten zeigen weiterhin die Übungen der Routine. Wenn man eine Routine ändert, hat dies also nur Einfluss auf Trainingseinheiten mit dem Status «geplant».

Abbruch: Eine Trainingseinheit, welche nicht aus dem Status «geplant» kommt, wird hart gelöscht. Man speichert nichts davon ab. Wird eine Routine während des Trainings gelöscht, wird auch eine geplante Einheit hart gelöscht.

Lücken in der Satznummerierung: Die Nummerierung der Sätze wird bei jedem Speichern neu gesetzt und gerendert. Wenn man jetzt 4 Sätze hat und man Satz 3 löscht, hat man die Sätze 1, 2 und 4. Diese werden direkt neu sortiert nach 1, 2 und 3. Die Nummerierung im Frontend bleibt bestehen.

Login-Sperre: Für die Login-Sperre wird eine eigene Tabelle geführt.

Zeitzone: Als Zeitzone wird Europe/Zurich genommen. DATE für das geplante Datum und TIMESTAMPTZ für präzisen Start und Abschluss bei den Trainingseinheiten.

Richtung der Abhängigkeiten zwischen den Paketen: Bei den Paketen sind zirkuläre Abhängigkeiten unbedingt zu vermeiden. Diese werden über Interfaces aufgelöst.

6.1 Datenmodell

Das Ziel des Datenmodells ist es, daraus eine einheitliche DDL festzulegen, welche bei der Entwicklung in eine PostgreSQL-Datenbank eingespielt wird. Tabellen, Spalten, Typen, NULL-Regeln, Keys und die Löschregeln sind finalisiert.

Die Namen sind englisch in snake_case. «User» und «Set» sind in PostgreSQL reservierte Wörter, deshalb heissen die Tabellen «app_user» und «exercise_set».

Die Tabellen «muscle_group» und «session_status» sind unveränderbare Listenwerte, die auch im Code benutzt werden. Sie gehören keinem Sportler. Die Werte enthalten auch eine feste ID. Somit kann der Code mit den festen IDs arbeiten.

Die Constraints und Indizes heissen nach dem Muster Präfix + Tabelle + Spalte. Die Präfixe sind pk_, fk_ uq_, ck_ und ix_.

6.1.1 ERM

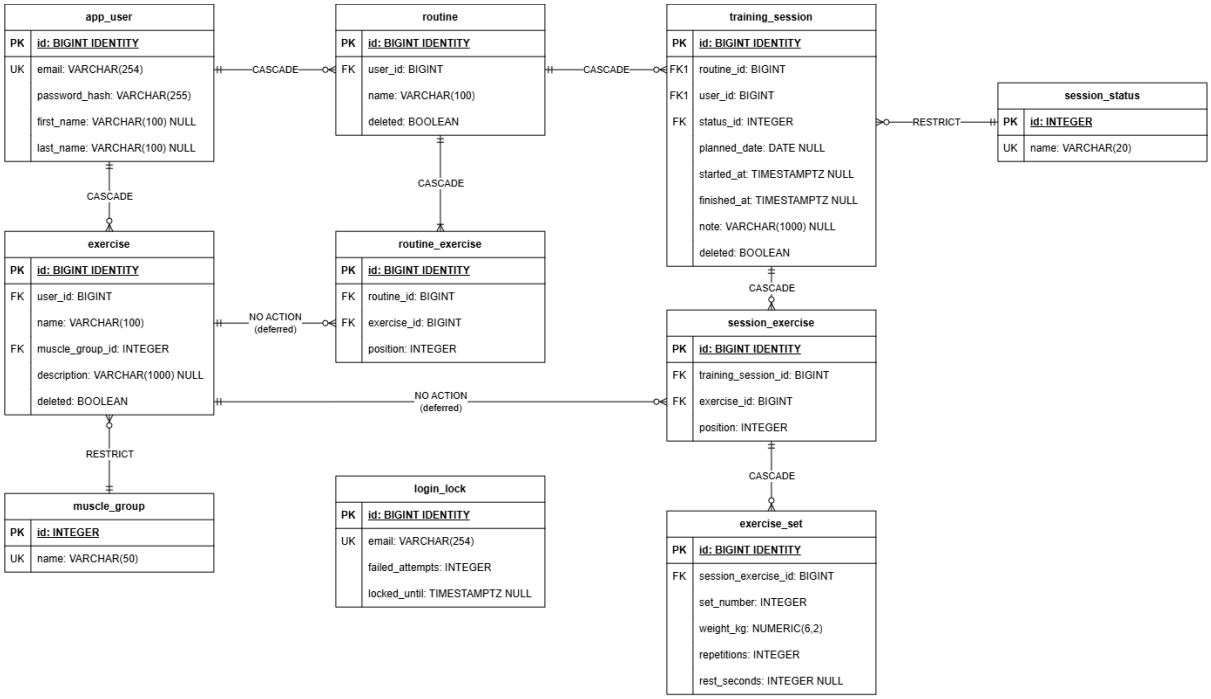


Abbildung 11: ERM

Das Attribut «training_session.user_id» lässt sich rein theoretisch von der Routine ableiten. Da aber eine Routine nie den Besitzer wechselt und man dafür direkt auf den Besitz der Trainingseinheit über user_id prüfen kann, ist das bewusst so gewählt.

6.1.2 Löschverhalten

Das Löschkonzept soll auf der DB durchgezogen werden. Für eine Kontolöschung reicht ein einziges DELETE auf die Tabelle app_user. Alle übrigen Daten werden über Regeln mitgelöscht. Löscht man eine Übung, Routine oder Trainingseinheit, wird ein Soft Delete ausgelöst (je nach Business Regel) über die Spalte «deleted».

Tabelle 70: Löschverhalten der Fremdschlüssel

Fremdschlüssel	ON DELETE	Begründung
exercise.user_id → app_user	CASCADE	Die Übungen des Sportlers werden beim Löschen des Benutzers mitgelöscht.
routine.user_id → app_user	CASCADE	Die Routinen des Sportlers werden beim Löschen des Benutzers mitgelöscht.
routine_exercise.routine_id → routine	CASCADE	Routineübungen gibt es nur innerhalb der Routine.
training_session.(routine_id, user_id) → routine(id, user_id)	CASCADE	Trainingseinheiten werden beim Löschen des Benutzers der Routine mitgelöscht.

session_exercise.training_session_id → training_session	CASCADE	Die kopierten Übungen gibt es nur innerhalb der Trainingseinheit.
exercise_set.session_exercise_id → session_exercise	CASCADE	Die Sätze gibt es nur innerhalb der Trainingseinheit.
routine_exercise.exercise_id → exercise	NO ACTION, DEFERRABLE INITIALLY DEFERRED	Eine Übung, die in einer Routineübung vorhanden ist, kann nicht hart gelöscht werden.
session_exercise.exercise_id → exercise	NO ACTION, DEFERRABLE INITIALLY DEFERRED	Eine Übung, die in einem Training vorhanden ist, kann nicht hart gelöscht werden.
training_session.status_id → session_status	RESTRICT	Der Statuseintrag wird nicht gelöscht.
exercise.muscle_group_id → muscle_group	RESTRICT	Die Muskelgruppe wird nicht gelöscht.

Die Regel NO ACTION DEFERRABLE INITIALLY DEFERRED greift folgendermassen. Die korrekte Erklärung wurde mit der Hilfe von Claude Opus 5.5 erarbeitet: [16]

NO ACTION: Die Regel löst selbst nichts aus. Es wird nichts mitgelöscht und nichts auf NULL gesetzt.

DEFERRABLE INITIALLY DEFERRED: Die Prüfung wird auf den COMMIT der Transaktion verschoben. Eine sofortige Prüfung würde die Löschung abbrechen, bevor die zweite Kaskade die verweisenden Zeilen entfernt hat.

6.1.3 Umsetzung der Löschung

Tabelle 71: Umsetzung der Löschung

Aktion	Umsetzung
Konto löschen	In einer Transaktion: DELETE auf app_user (der Rest wird über die Fremdschlüssel mitgelöscht) und DELETE auf login_lock für die E-Mail-Adresse.
Übung löschen	• Eine nicht gelöschte Routine mit dieser Übung blockiert das Löschen komplett. • Übung wird soft deleted
Routine löschen	• Trainingseinheiten mit dem Status «geplant» werden hart gelöscht. An diesen hängen auch noch keine Trainingssätze. • Routine wird soft deleted.
Trainingseinheit	• Eine geplante Trainingseinheit wird hart gelöscht. • Eine abgeschlossene wird soft gelöscht. • Eine aktive kann man nur abbrechen.
Training abbrechen	Eine spontane Trainingseinheit hat kein planned_date und wird somit hart gelöscht. Sobald es ein planned_date hat, werden die session_exercise hart gelöscht, der Status auf «geplant» gesetzt und das started_at auf NULL gesetzt. Ist die Routine während dem Training gelöscht worden, wird die Trainingseinheit hart gelöscht.

Satz löschen	Sätze werden immer hart gelöscht. Anschliessend wird set_number neu nummeriert.
Training abschliessen	Übungen, die keinen Trainingssatz haben, werden hart gelöscht.

6.1.4 Eindeutigkeit

Tabelle 72: Eindeutigkeit

Tabelle	Eindeutig über
app_user	email
login_lock	email
muscle_group	name
session_status	name
exercise	user_id, lower(name), nur nicht gelöschte Zeilen
routine	user_id, lower(name), nur nicht gelöschte Zeilen
routine	id, user_id
routine_exercise	routine_id, exercise_id *(verzögert)*
routine_exercise	routine_id, position *(verzögert)*
session_exercise	training_session_id, exercise_id *(verzögert)*
session_exercise	training_session_id, position *(verzögert)*
exercise_set	session_exercise_id, set_number *(verzögert)*
trainings_session	user_id, nur bei status = 2 (aktiv)

Der Namen einer Übung oder Routinen kann nach einem Soft Delete wiederverwendet werden.

Die Spalten mit *(verzögert)* sind DEFERRABLE INITIALLY DEFERRED. Beim Verschieben einer Übung oder Neunummerieren können zwei Zeilen kurzzeitig dieselbe Nummer haben, bevor die Transaktion abgeschlossen wird.

6.1.5 Prüfregeln

Tabelle 73: Prüfregeln

Tabelle	Regel
app_user, login_lock	email = lower(email)
login_lock	failed_attempts ≥ 0
routine_exercise, session_exercise	position ≥ 1
exercise_set	set_number ≥ 1
exercise_set	weight_kg zwischen 0 und 1000, Typ NUMERIC(6,2)
exercise_set	repetitions zwischen 1 und 100
exercise_set	rest_seconds zwischen 0 und 3600 oder leer
training_session	deleted nur bei Status «abgeschlossen»

6.1.6 Indizes

Bei PostgreSQL erhalten Primärschlüssel und UNIQUE-Constraints automatisch einen Index. Für Fremdschlüssel nicht. Fremdschlüssel, die bereits in einem UNIQUE-Constraint ganz vorne stehen, erhalten deshalb keinen weiteren Index. Dies betrifft folgende:

- routine_exercise.routine_id
- session_exercise.training_session_id
- exercise_set.session_exercise_id

Verweise auf Stammdaten enthalten auch keinen Index, da es nur wenige feste Werte gibt, welche nie gelöscht werden.

Somit werden folgende Indizes angelegt:

Tabelle 74: Indizes

Index	Tabelle (Spalte)
ix_exercise_user_id	exercise (user_id)
ix_routine_user_id	routine (user_id)
ix_routine_exercise_exercise_id	routine_exercise (exercise_id)
ix_training_session_routine_id	training_session (routine_id)
ix_session_exercise_exercise_id	session_exercise (exercise_id)

6.1.7 Stammdatenwerte

Die Werte werden in der ersten Version mit festen IDs eingefügt.

Muskelgruppen:

Tabelle 75: Stammdaten Muskelgruppen

muscle_group.id	name
1	Brust
2	Rücken
3	Schultern
4	Bizeps
5	Trizeps
6	Bauch
7	Beine
8	Gesäss

Trainingseinheit Status:

Tabelle 76: Stammdaten Status

session_status.id	name
1	geplant
2	aktiv
3	abgeschlossen

6.2 Backend-Struktur

Die Backendstruktur ist grob vorgegeben von der Entscheidung des Backend-Stils (E2). Es wird das Konzept des Schichtenmonolithen ausgearbeitet. Mit dem gegebenen Datenmodell hat man eine Basis, auf der man aufbauen kann. Während der Entwicklung werden die einzelnen Methoden, Attribute und Beziehungen dazukommen.

6.2.1 Klassendiagramme

6.2.1.1 Paketdiagramm

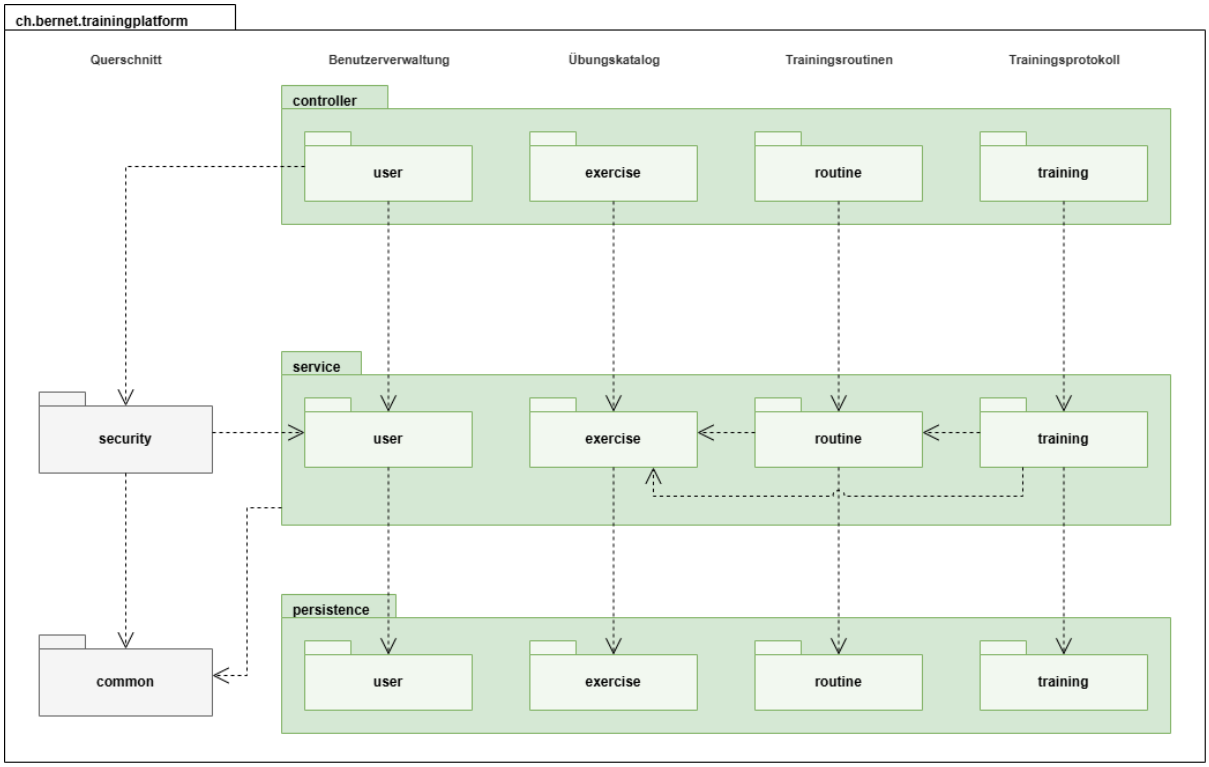


Abbildung 12: Paketdiagramm

6.2.1.2 Benutzerverwaltung

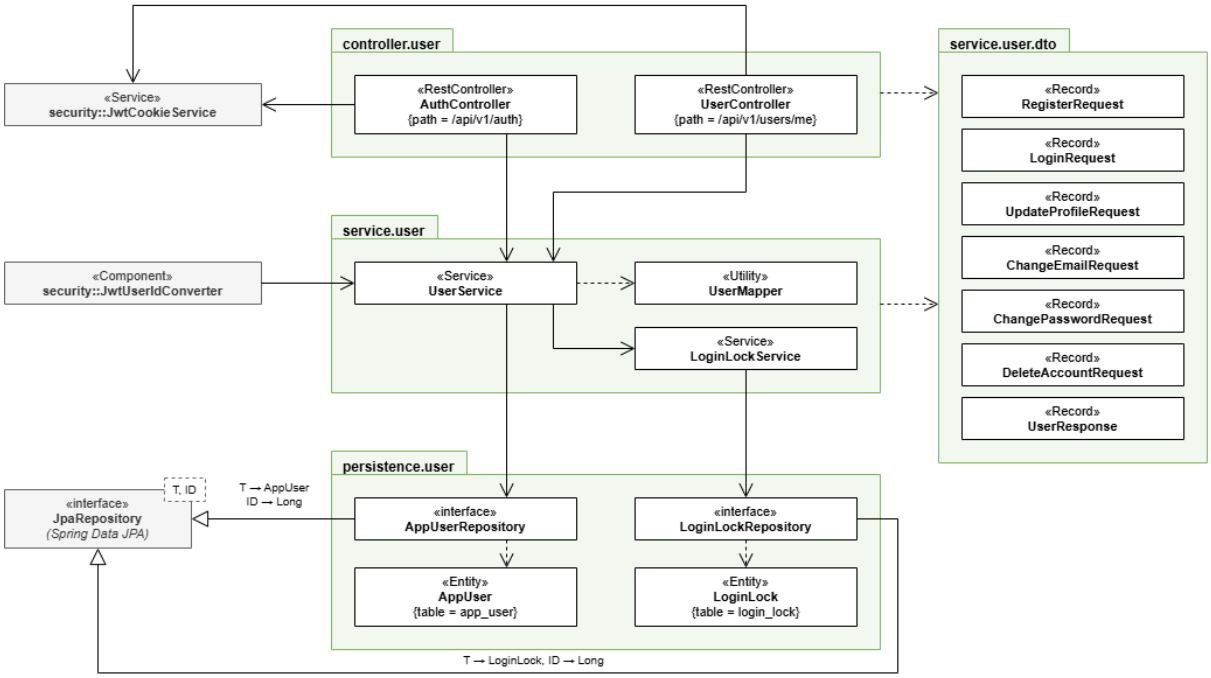


Abbildung 13: Klassendiagramm Benutzerverwaltung

6.2.1.3 Übungskatalog

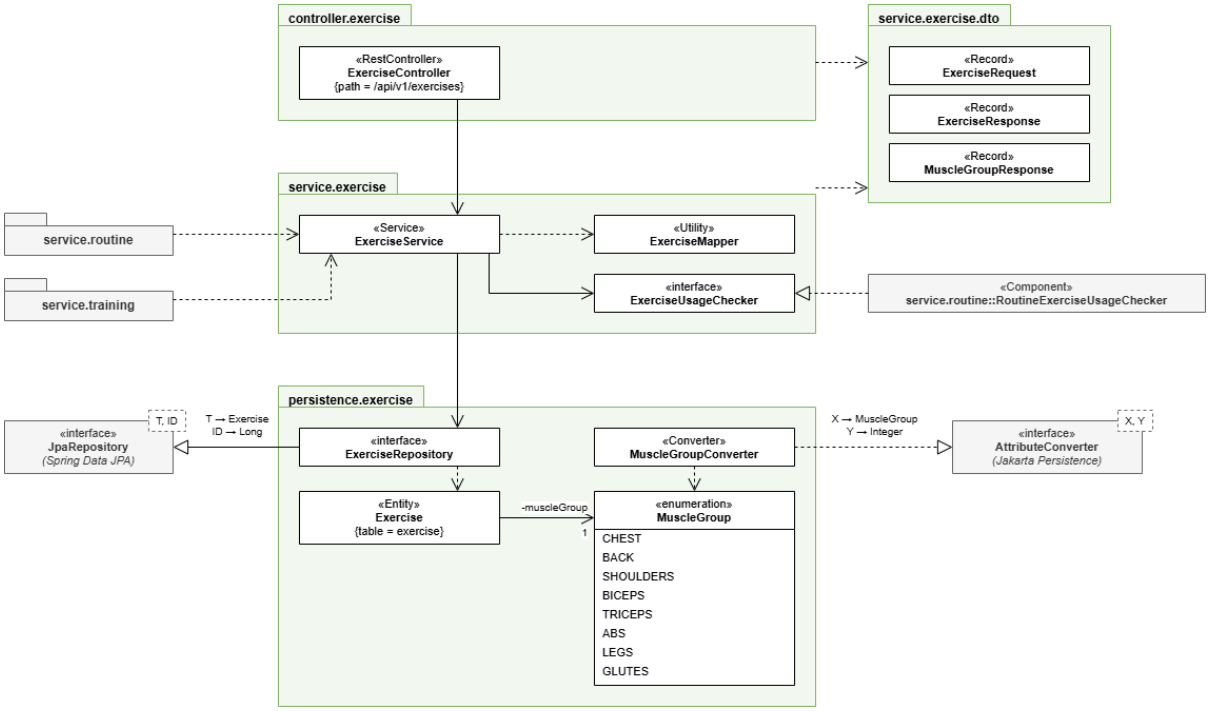


Abbildung 14: Klassendiagramm Übungskatalog

6.2.1.4 Trainingsroutinen

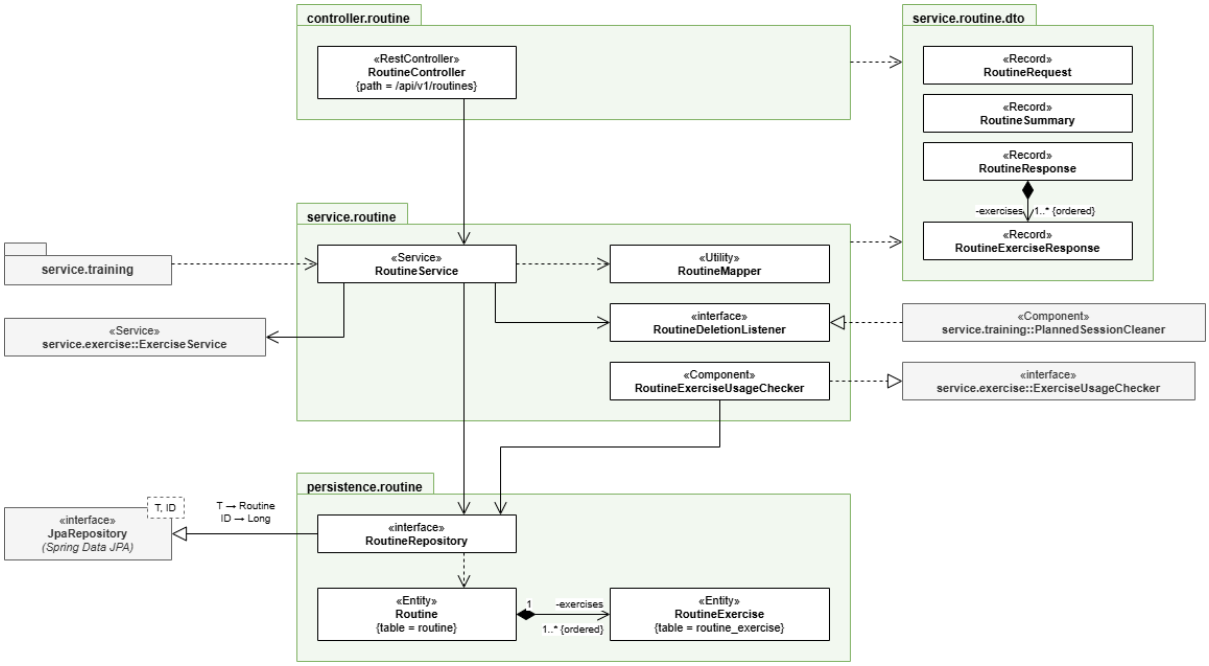


Abbildung 15: Klassendiagramm Trainingsroutinen

6.2.1.5 Trainingsprotokoll Controller und Service

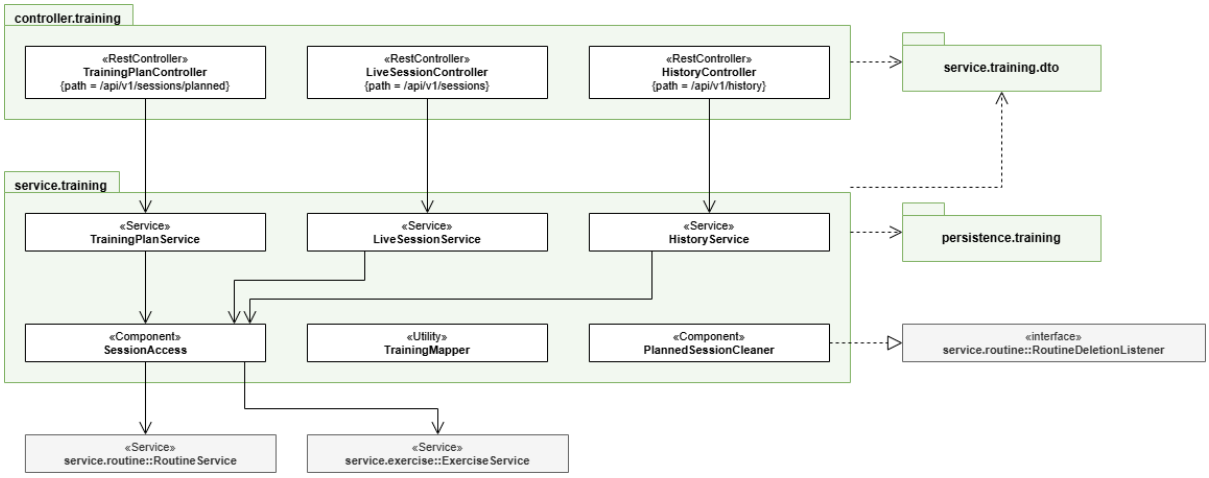


Abbildung 16: Klassendiagramm Trainingsprotokoll Controller und Service

6.2.1.6 Trainingsprotokoll DTOs

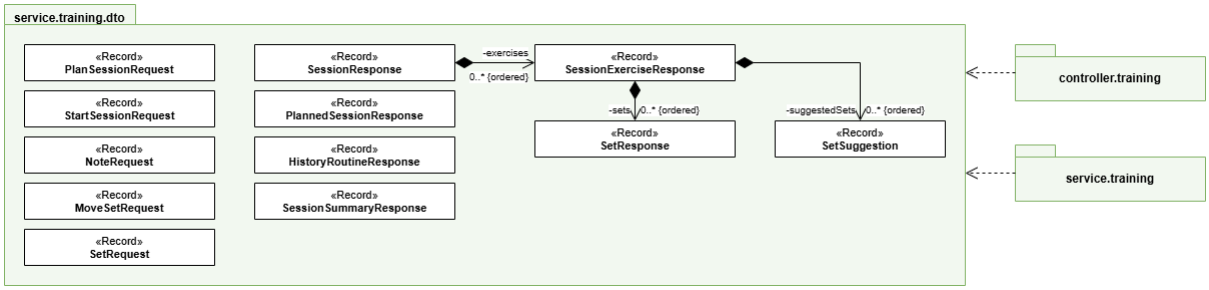


Abbildung 17: Klassendiagramm Trainingsprotokoll DTOs

6.2.1.7 Trainingsprotokoll Persistenz

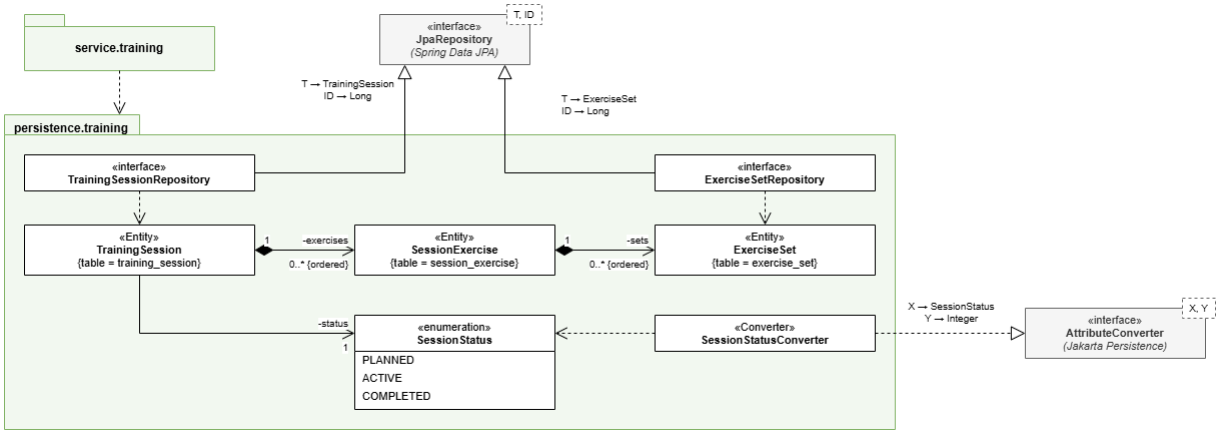


Abbildung 18: Klassendiagramm Trainingsprotokoll Persistenz

6.2.1.8 Querschnitt

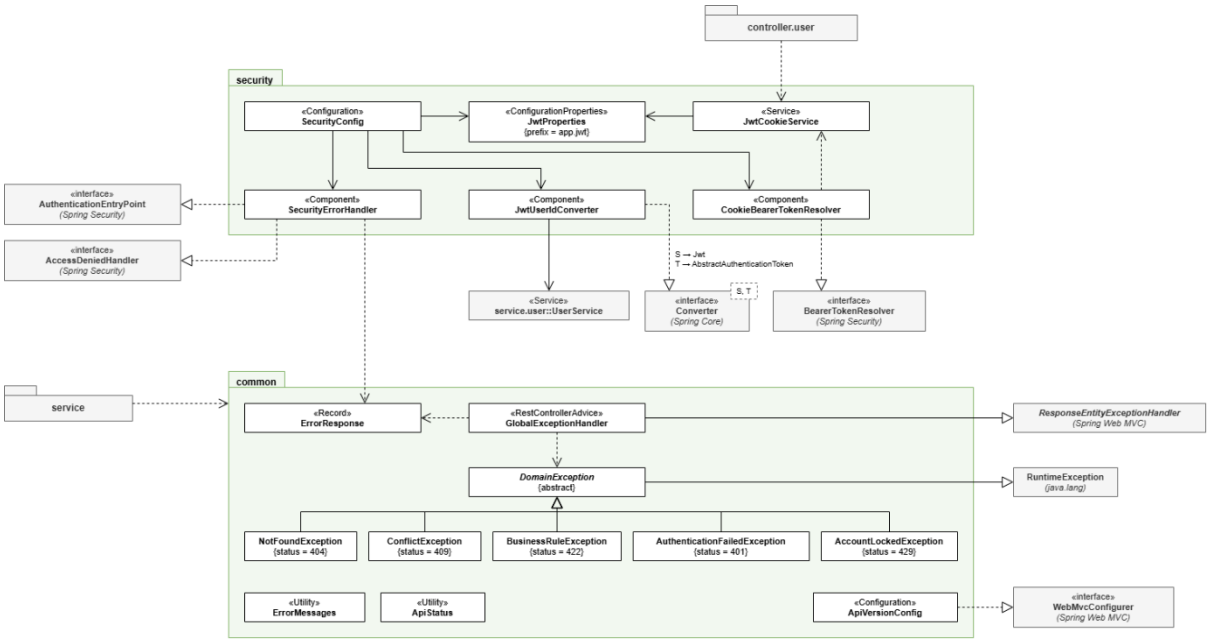


Abbildung 19: Klassendiagramm Querschnitt

6.2.2 Business Regeln

Manche Regeln, welche es zwar auch schon auf der Datenbank gibt, werden trotzdem vorher im Backend auch nochmals geprüft, um eine saubere Fehlermeldung zu generieren.

Jede Regel gehört zu genau einer Stufe:

1. Controller: Pflichtfelder, Formate und Wertebereiche der Request-DTOs mit Bean Validation. Fehler erzeugen einen 400-Fehlercode.
2. Service: Regeln, die vom Datenbestand, vom Status oder aktuellen Datum abhängen. Fehler erzeugen Fehlercode 401, 404, 409, 422 oder 429.
3. Datenbank: NOT NULL, UNIQUE, CHECK und Fremdschlüssel als absolut letzte Absicherung. Fehler erzeugen einen 500-Fehlercode und sollten in der Produktion eigentlich nicht auftreten.

6.2.2.1 Regeln im Service

Die Regeln im Service sind zahlreich. Darum werden diese in einer Tabelle genauer aufgezeigt.

Tabelle 77: Regeln im Service

Regel	Fehler
Benutzerverwaltung	
Die E-Mail-Adresse wird getrimmt und klein geschrieben gespeichert. Pro Adresse gibt es höchstens ein Konto.	409
Beim Login erhalten eine unbekannte E-Mail-Adresse und ein falsches Passwort dieselbe Antwort.	401
Nach 5 Fehlversuchen in Folge ist die E-Mail-Adresse 15 Minuten gesperrt, auch wenn es dazu kein Konto gibt.	429

E-Mail-Adresse ändern, Passwort ändern und Konto löschen verlangen das aktuelle Passwort. Fehlversuche zählen auch bei der Passwortbestätigung und können eine Login-Sperre verursachen.	401
Übungskatalog	
Der Name ist je Sportler eindeutig, ohne Unterscheidung von Gross- und Kleinschreibung. Soft-gelöschte Übungen zählen nicht.	409
Eine Übung kann erst soft-gelöscht werden, wenn sie in keiner nicht gelöschten Routine mehr vorkommt.	409
Trainingsroutinen	
Der Name ist eindeutig, nach derselben Regel wie bei den Übungen.	409
Jede Übung kommt höchstens einmal vor. Die Positionen vergibt das System lückenlos ab 1, in der Reihenfolge des Requests.	422
Alle Übungen müssen aktive, eigene Übungen sein	404
Live-Protokollierung	
Das Datum darf nicht vor Heute liegen.	422
Sätze lassen sich nur in der aktiven Einheit erfassen, ändern, verschieben und löschen.	404
Die Satznummern sind lückenlos ab 1, auch nach dem Löschen oder Verschieben eines Satzes. Die Zielposition muss innerhalb der vorhandenen Sätze liegen.	422
Vorgeschlagen werden die Sätze der letzten abgeschlossenen, nicht gelöschten Einheit derselben Routine, in der die Übung vorkam. Offen sind die Sätze mit einer Satznummer grösser als die Anzahl bereits erfasster Sätze. Danach oder ohne Historie bleibt die Liste leer. Im Verlauf ist sie immer leer.	–

6.2.2.2 Statusmodell der Trainingseinheit

Den Status einer Einheit ändern nur die Methoden der Trainingseinheit. So prüft jeder Übergang immer die Bedingung, egal welcher Service ihn auslöst.

Tabelle 78: Statusmodell der Trainingseinheit

Aktion	Status	Bedingung (Fehler)	Wirkung
Einplanen	→ Geplant	Datum heute oder später (422)	Routine und planned_date setzen
Ändern	Geplant	Nur im Status «geplant» (404) Datum heute oder später (422)	Routine und planned_date ersetzen
Starten	Geplant → aktiv	Keine andere Einheit aktiv (409)	Übungen der Routine kopieren, started_at setzen

Spontan starten	→ Aktiv	Keine andere Einheit aktiv (409)	Wie Starten, aber ohne planned_date
Abschliessen	Aktiv → Abgeschlossen	Mindestens ein Satz (409)	Übungen ohne Satz löschen, finished_at setzen
Abbrechen	Aktiv → Geplant	–	Kopierte Übungen und Sätze löschen, started_at leeren.
Abbrechen	Aktiv -> Hart gelöscht		Eine spontane Einheit wird ganz gelöscht.
Abbrechen	Geplant -> Hart gelöscht	-	Wenn während einer geplanten Trainingseinheit die Routine gelöscht wird, und man das Training abbricht, wird die Trainingseinheit komplett gelöscht.
Routine gelöscht	Geplant -> Hart gelöscht	-	Die Trainingseinheit wird hart gelöscht, da es die Routine nicht mehr gibt.
Aus Verlauf löschen	Abgeschlossen -> Deleted	Trainingseinheit muss abgeschlossen sein (404)	Die Trainingseinheit wird aus dem Verlauf gelöscht.

Daraus ergeben sich folgende Zeitstempel pro Status:

- «geplant» hat ein planned_date, aber weder started_at noch finished_at
- «aktiv» hat ein «started_at» und ein planned_date, wenn es keine spontane Trainingseinheit ist.
- «abgeschlossen» hat mindestens finished_at und started_at

6.3 REST-Schnittstelle

Das Backend stellt seine gesamte Funktionalität über eine REST-Schnittstelle bereit. Das Frontend greift ausschliesslich über HTTP mit JSON-Daten darauf zu. Dieses Kapitel gibt eine Übersicht aller Endpoints, die übertragenen Datenobjekte, Fehlerformate und die Authentifizierung.

6.3.1 Grundsätze

- Alle Endpunkte liegen unter /api/v1. Darunter ist die Schnittstelle je nach Fachbereich aufgeteilt. Jeder Fachbereich entspricht einem Controller.
- Es gibt eine API-Versionierung. Eine unbekannte Version ergibt 400, ohne Version gibt es 404.
- Aufteilung der HTTP-Methoden: GET liest, POST legt an, PUT ersetzt vollständig und DELETE löscht
- Ressourcen, die es pro Sportler nur einmal gibt, haben einen festen Pfad. Z.B. /api/v1/users/me. Der Benutzer wird über die Authentifizierung ermittelt.
- Die Benutzer-ID wird nie aus dem Pfad oder dem Body gelesen, sondern immer aus dem geprüften Token der Authentifizierung. Alle Services filtern jeweils nach dieser ID. Ein Sportler kann somit nur auf seine eigenen Daten zugreifen.

6.3.2 Statuscodes

Tabelle 79: Statuscodes

Code	Bedeutung in dieser Schnittstelle
200 OK	Erfolgreich, die Antwort enthält einen Body
201 Created	Ressource angelegt, der Body enthält die neue Ressource
204 No Content	Erfolgreich, die Antwort enthält keinen Body
400 Bad Request	Das Request-DTO verletzt eine Validierungsregel
401 Unauthorized	Kein oder ungültiges Token bzw. falsche Zugangsdaten
404 Not Found	Die Ressource existiert nicht oder gehört einem anderen Sportler
409 Conflict	Widerspruch zum gespeicherten Datenbestand, z.B. ein bereits vergebener Name
422 Unprocessable Content	Fachliche Regel verletzt, die sich aus dem Request allein ergibt, z.B. ein Datum in der Vergangenheit
429 Too Many Requests	Login nach zu vielen Fehlversuchen vorübergehend gesperrt
500 Internal Server Error	Unerwarteter Fehler auf dem Server

6.3.3 Endpunkte

Tabelle 80: Endpunkte

Methode	Pfad	Zweck	Statuscodes
Authentifizierung			
POST	/api/v1/auth/register	Konto anlegen und anmelden	201, 400, 409
POST	/api/v1/auth/login	Anmelden	200, 400, 401, 429
POST	/api/v1/auth/logout	Abmelden, Cookie löschen	204
GET	/api/actuator/health	Health-Check	200, 503
Benutzerkonto			
GET	/api/v1/users/me	Eigenes Profil lesen	200

PUT	/api/v1/users/me	Vor- und Nachname ändern	200, 400
PUT	/api/v1/users/me/email	E-Mail-Adresse ändern	200, 400, 401, 409
PUT	/api/v1/users/me/password	Passwort ändern	204, 400, 401
DELETE	/api/v1/users/me	Konto mit allen Daten löschen	204, 400, 401
Übungen			
GET	/api/v1/exercises	Eigene Übungen auflisten	200
GET	/api/v1/exercises/muscle-groups	Muskelgruppen auflisten	200
POST	/api/v1/exercises	Übung anlegen	201, 400, 409
PUT	/api/v1/exercises/{exerciseId}	Übung ändern	200, 400, 404, 409
DELETE	/api/v1/exercises/{exerciseId}	Übung löschen	204, 404, 409
Routinen			
GET	/api/v1/routines	Eigene Routinen auflisten	200
GET	/api/v1/routines/{routineId}	Routine mit Übungen lesen	200, 404
POST	/api/v1/routines	Routine anlegen	201, 400, 404, 409, 422
PUT	/api/v1/routines/{routineId}	Routine ändern	200, 400, 404, 409, 422
DELETE	/api/v1/routines/{routineId}	Routine löschen	204, 404
Trainingsplanung			
GET	/api/v1/sessions/planned	Geplante Einheiten auflisten	200

POST	/api/v1/sessions/planned	Einheit planen	201, 400, 404, 422
PUT	/api/v1/sessions/planned/{sessionId}	Geplante Einheit verschieben	200, 400, 404, 422
DELETE	/api/v1/sessions/planned/{sessionId}	Geplante Einheit löschen	204, 404
Live-Training			
POST	/api/v1/sessions/planned/{sessionId}/start	Geplante Einheit starten	200, 404, 409
POST	/api/v1/sessions	Spontane Einheit starten	201, 400, 404, 409
GET	/api/v1/sessions/active	Aktive Einheit lesen	200, 204
PUT	/api/v1/sessions/active/note	Notiz zur Einheit setzen	204, 400, 404
POST	/api/v1/sessions/active/exercises/{sessionExerciseId}/sets	Satz erfassen	201, 400, 404
PUT	/api/v1/sessions/active/sets/{setId}	Satz ändern	200, 400, 404
PUT	/api/v1/sessions/active/sets/{setId}/position	Satz verschieben	200, 400, 404, 422
DELETE	/api/v1/sessions/active/sets/{setId}	Satz löschen	200, 404
POST	/api/v1/sessions/active/finish	Einheit abschliessen	204, 404, 409
POST	/api/v1/sessions/active/cancel	Einheit abbrechen	204, 404
Verlauf			
GET	/api/v1/history/routines	Routinen mit abgeschlossenen Einheiten auflisten	200
GET	/api/v1/history/routines/{routineId}/sessions	Abgeschlossene Einheiten einer Routine auflisten	200

GET	/api/v1/history/sessions/{sessionId}	Abgeschlossene Einheit im Detail lesen	200, 404
DELETE	/api/v1/history/sessions/{sessionId}	Einheit aus dem Verlauf löschen	204, 404

6.3.4 Datenübertragungsobjekte

Die JPA-Entitäten verlassen die Service-Schicht nie. Controller nehmen ausschliesslich Request-DTOs entgegen und geben Response-DTOs zurück. Alle DTOs werden als Java-Records umgesetzt. Diese Trennung hat mehrere Vorteile. Das Datenbankschema kann sich ändern, ohne dass sich die Schnittstelle ändert. Sensible Felder wie das Passwort gelangen nicht versehentlich in eine Antwort. Es entstehen keine zyklischen Referenzen zwischen den Entitäten und Records kommen ohne Boilerplate-Code aus.

6.3.4.1 Request-DTOs

Tabelle 81: Request-DTOs

DTO	Felder	Validierung	Verwendung
RegisterRequest	email password	email: Pflicht, E-Mail-Format, max. 254 Zeichen password: Pflicht, Passwortrichtlinie (15-64 Zeichen)	POST /auth/register
LoginRequest	email password	email: Pflicht, E-Mail-Format, max. 254 Zeichen password: Pflicht, Passwortrichtlinie (15-64 Zeichen)	POST /auth/login
UpdateProfileRequest	firstName lastName	Optional, je max. 100 Zeichen	PUT /users/me
ChangeEmailRequest	newEmail password	newEmail: Pflicht, E-Mail-Format, max. 254 Zeichen password: Pflicht, Passwortrichtlinie (15-64 Zeichen)	PUT /users/me/email
ChangePasswordRequest	currentPassword newPassword	Beide Pflicht newPassword: Passwortrichtlinie (15-64 Zeichen)	PUT /users/me/password

DeleteAccountRequest	password	Pflicht	DELETE /users/me
ExerciseRequest	name muscleGroupId description	name: Pflicht, max. 100 Zeichen muscleGroupId: zwischen 1 und 8 description: optional, max. 1000 Zeichen	POST, PUT /exercises
RoutineRequest	name exerciseIds	name: Pflicht, max. 100 Zeichen exerciseIds: mind. ein Eintrag, Reihenfolge = Position	POST, PUT /routines
PlanSessionRequest	routineId plannedDate	Beide Pflicht	POST, PUT /sessions/planned
StartSessionRequest	routineId	Pflicht	POST /sessions
NoteRequest	note	max. 1000 Zeichen	PUT /sessions/active/note
SetRequest	weightKg repetitions restSeconds	weightKg: Pflicht, 0–1000, max. 2 Nachkommastellen repetitions: 1–100, pflicht restSeconds: optional, 0–3600	Satz erfassen und ändern
MoveSetRequest	newSetNumber	Mind. 1	Satz verschieben

6.3.4.2 Response-DTOs

Tabelle 82: Response-DTOs

DTO	Felder	Verwendung
UserResponse	id, email, firstName, lastName	Authentifizierung, Benutzerkonto
MuscleGroupResponse	id, name	Muskelgruppen
ExerciseResponse	id, name, muscleGroupId, muscleGroupName, description	Übungen
RoutineResponse	id, name, exercises (Liste von RoutineExerciseResponse)	Routinen
RoutineExerciseResponse	position, exerciseId, name, muscleGroupName	In RoutineResponse
PlannedSessionResponse	id, routineId, routineName, plannedDate	Trainingsplanung
SessionResponse	id, routineId, routineName, routineDeleted, status, startedAt, finishedAt, note, exercises (Liste von SessionExerciseResponse)	Live-Training, Detail im Verlauf
SessionExerciseResponse	id, exerciseId, exerciseName, position, sets (Liste von SetResponse), suggestedSets (Liste von SetSuggestion, im Verlauf leer)	Satz-Endpunkte, in SessionResponse
SetResponse	id, setNumber, weightKg, repetitions, restSeconds	In SessionExerciseResponse
SetSuggestion	setNumber, weightKg, repetitions, restSeconds	In SessionExerciseResponse
HistoryRoutineResponse	routineId, name, deleted	Verlauf: Routinen
SessionSummaryResponse	id, startedAt, finishedAt	Verlauf: Einheiten einer Routine

6.3.5 Einheitliches Fehlerformat

Jeder Fehler wird mit derselben JSON-Struktur ausgegeben. Das Frontend kann die Fehler dadurch an einer einzigen Stelle auswerten.

Tabelle 83: Fehlerformat

Feld	Inhalt
status	HTTP-Statuscode. Er ist redundant zum Header, damit der Body auch allein auswertbar ist.
message	Es ist eine englische Konstante aus ErrorMessages und das Frontend zeigt sie nie an. Im Frontend wird jeweils ein deutscher Text je Statuscode angezeigt.
details	Einzelfehler. Bei Validierungsfehlern steht ein Eintrag pro fehlerhaftem Feld, sonst ist die Liste leer.

6.4 Abläufe

6.4.1 Aktivitätsdiagramme

6.4.1.1 SAF7 Training einplanen

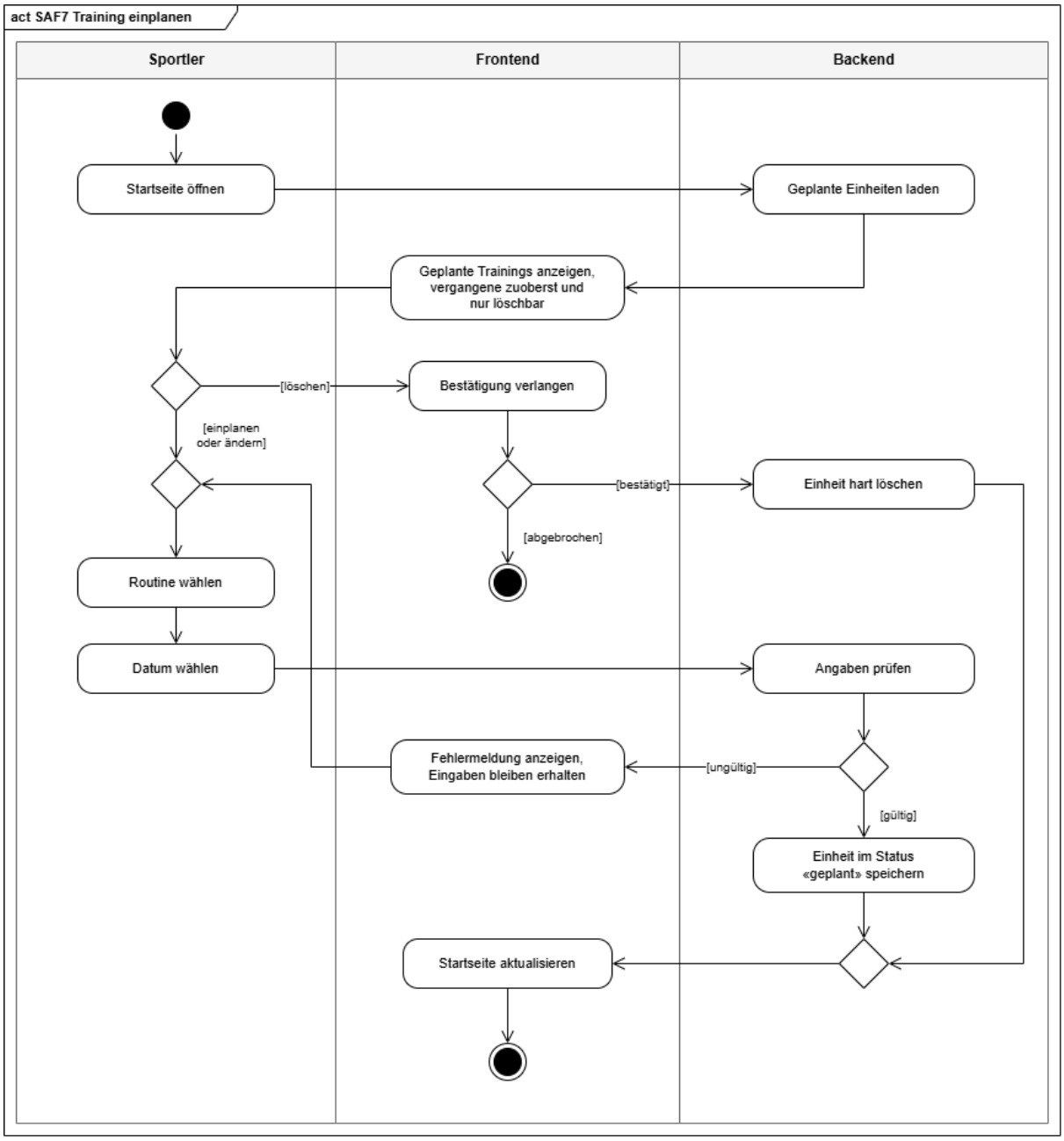


Abbildung 20: Aktivitätsdiagramm Training einplanen

6.4.1.2 SAF8 Training durchführen und protokollieren

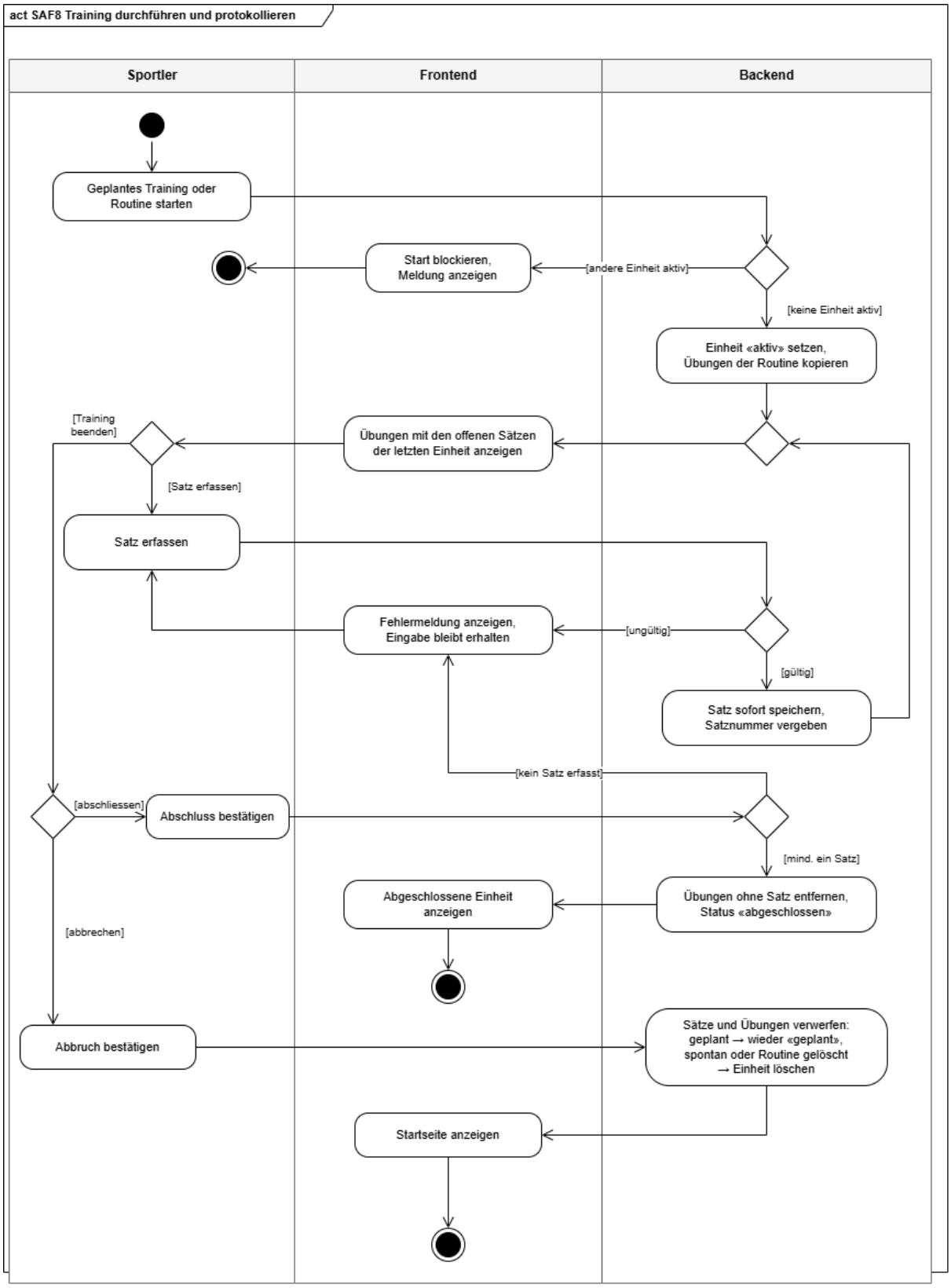


Abbildung 21: Aktivitätsdiagramm Training durchführen

6.4.2 Sequenzdiagramme

6.4.2.1 Login

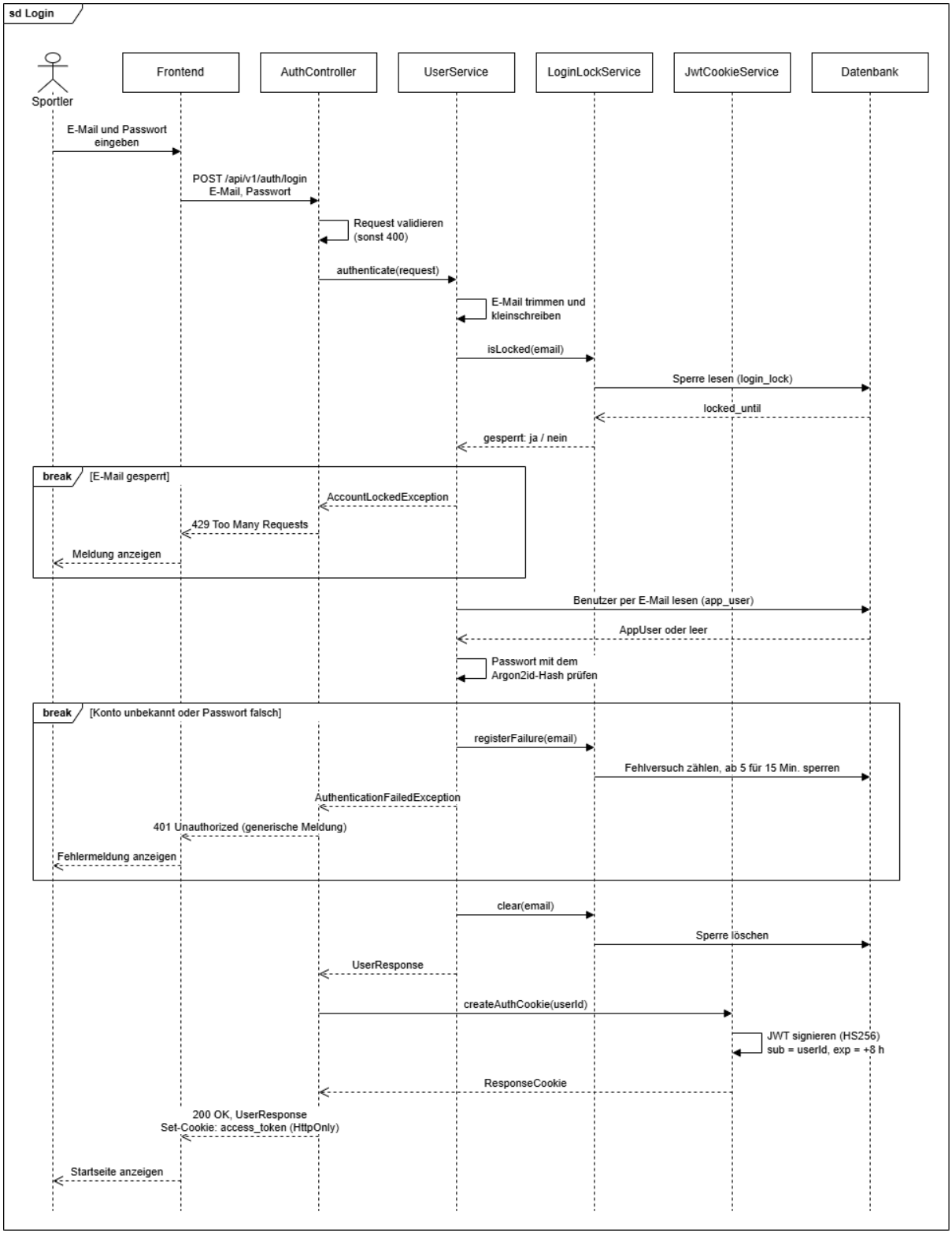


Abbildung 22: Sequenzdiagramm Login

6.4.2.2 Anfrage mit Token

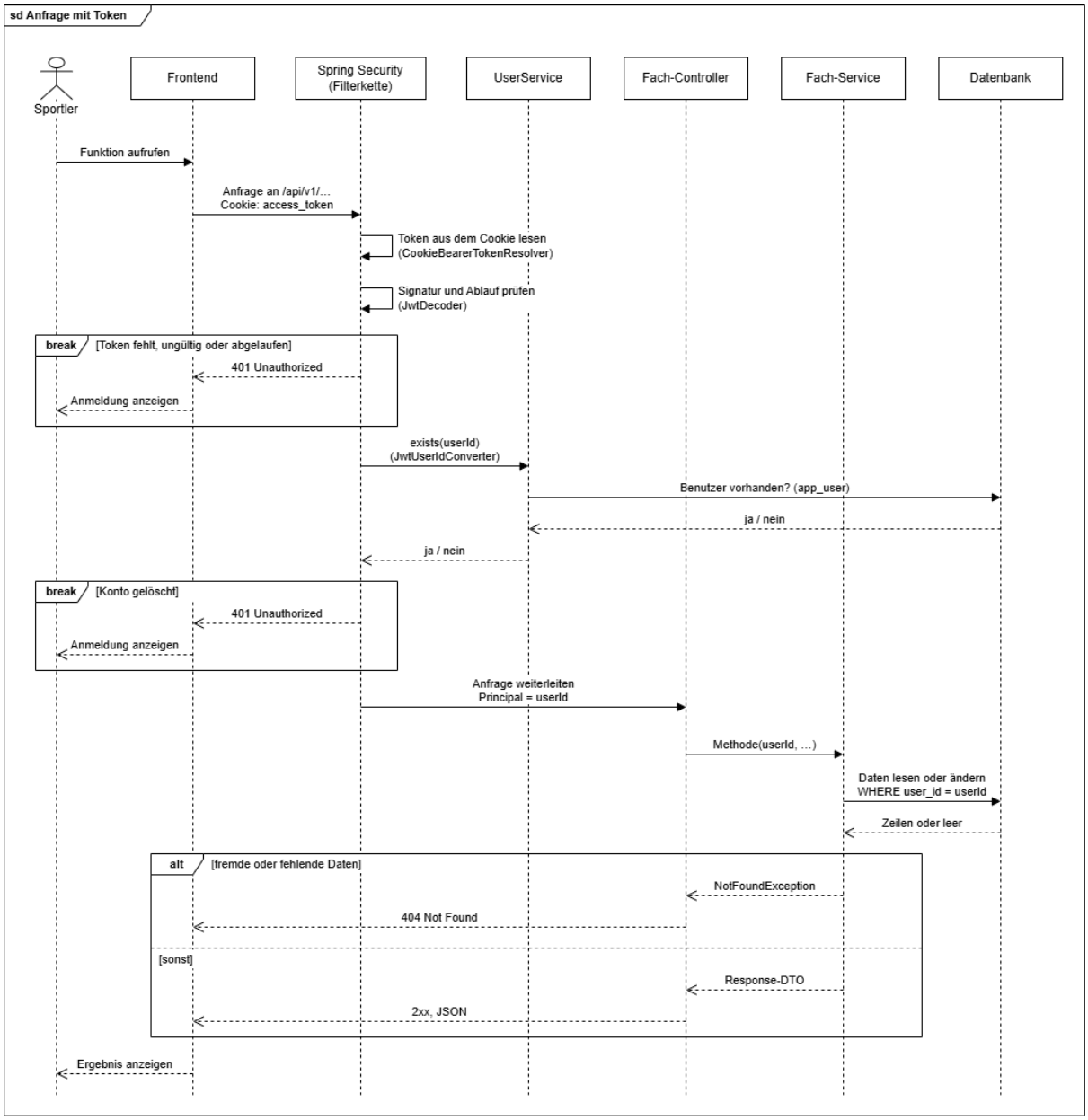


Abbildung 23: Sequenzdiagramm Anfrage mit Token

6.4.3 Zustandsdiagramm Status Trainingseinheit

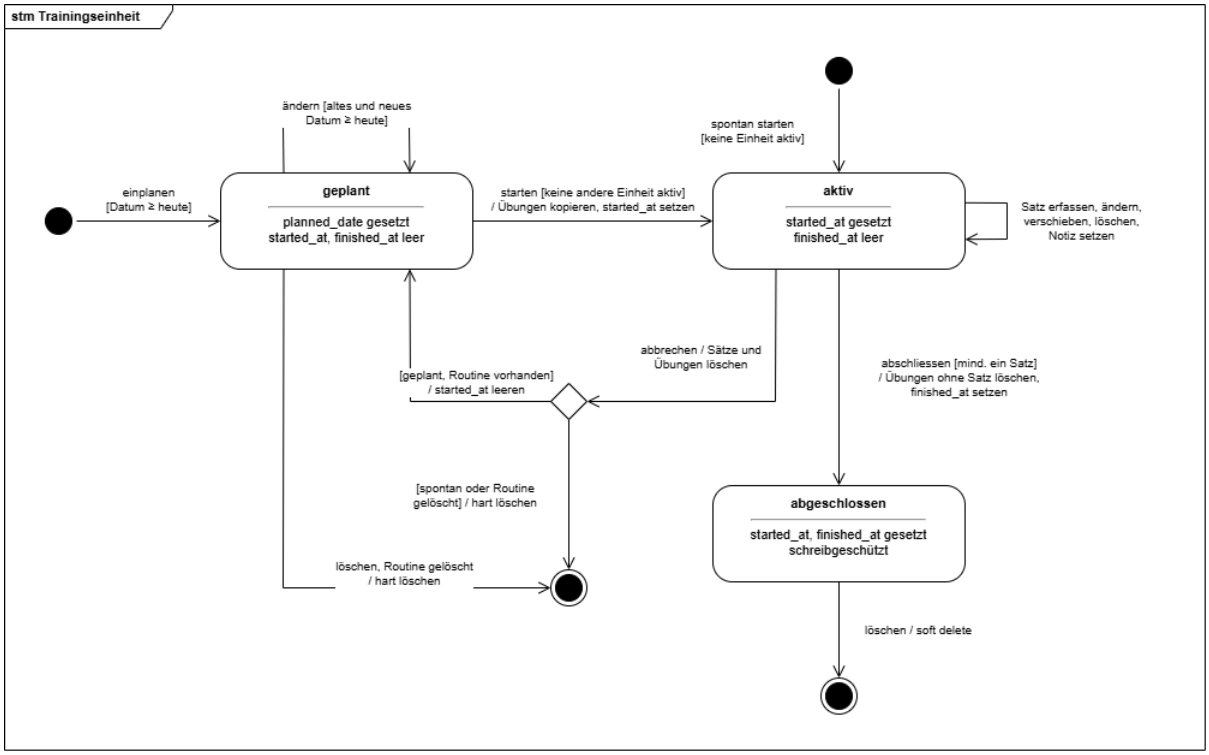


Abbildung 24: Zustandsdiagramm Trainingseinheit

6.5 GUI-Konzept

6.5.1 Navigation

Aus den Systemanwendungsfällen ergeben sich folgende Navigationen:

- **Öffentlich:** Das Anmelden, Registrieren und der Datenschutz können ohne Anmeldung geöffnet werden.
- **Hauptnavigation:** Es gibt am unteren Rand eine Leiste mit vier Bereichen. Start, Routinen, Übungen und Verlauf. Ab der Grösse eines Tablets wird es zu einer Seitenleiste. Oben rechts hat man einen Absprung auf die Ansicht des Kontos.
- **Livetraining:** Eine Vollbildansicht ohne Hauptnavigation.

Sobald man angemeldet ist, landet man auf dem Start der Hauptnavigation. Ansonsten wird man direkt auf den Anmelde-Screen weitergeleitet (z.B. wenn ein Token abläuft).

6.5.2 Screens

Tabelle 84: Screens

Screen	Pfad	Wichtigste Elemente
Anmelden	/anmelden	Anmeldemaske mit E-Mail und Passwort. Links zu Registrieren und Datenschutz.
Registrieren	/registrieren	Registrierungsmaske mit E-Mail, Passwort, Vorname und Nachname.

		Link zum Datenschutz.
Datenschutz	/datenschutz	Statischer Text, ohne Anmeldung erreichbar (#FA-1.4)
Start	/	Laufendes Training steht zuoberst mit «Fortsetzen». Knöpfe «Einplanen» und «Spontan starten» Liste der geplanten Trainings mit Starten und Ändern. Nur das nächste geplante Training hat einen gefüllten «Starten»-Knopf. Vergangene geplante Trainings sind als «vergangen» markiert. Geplante Trainings lassen sich löschen. «Spontan starten» zeigt die Routinen als Liste, ein Tipp startet direkt.
Training einplanen, Ändern	Dialog	Routine auswählen, Datum ab heute. Derselbe Dialog dient zum Ändern.
Spontan starten	Dialog	Routine auswählen, welche man für das spontane Training will.
Live-Training	/training	Die Kopfzeile zeigt Startzeit und laufende Dauer und einen «Zurück zum Start», das Training läuft dabei weiter. Pro Übung gibt es ein Akkordeon mit dem Beschreibungshinweis und dem Fortschritt «x von y Sätzen» mit Häkchen. Das Eingabeformular ist mit dem nächsten offenen Vorschlag vorbelegt. Die übrigen stehen darunter. Antippen eines Satzes öffnet einen Dialog: Werte ändern, Position ändern, Löschen mit Rückfrage. Die Notiz speichert sich beim Verlassen des Felds.
Routinen	/routinen	Liste der Routinen mit Anzahl Übungen, Button «Neue Routine»
Routine bearbeiten	/routinen/:id	Übungen werden über einen Dialog «Übungen hinzufügen» gewählt. Er erlaubt

		Mehrfachauswahl und hat ab 8 Übungen ein Suchfeld. Die ausgewählten Übungen kann man «nach oben» oder «nach unten» verschieben. Beim Löschen einer Routine erscheint ein Hinweis, dass geplante Trainings mitgelöscht werden.
Übungen	/uebungen	Alphabetische Liste mit Muskelgruppe, Knopf «Neue Übung» Es wird die Beschreibung auf 2 Zeilen gekürzt angezeigt.
Übung bearbeiten	Dialog	Name, Muskelgruppe (Auswahl), Beschreibung, Löschen
Verlauf	/verlauf	Routinen mit abgeschlossenen Einheiten, gelöschte Routinen markiert
Einheiten einer Routine	/verlauf/:routineId	Abgeschlossene Einheiten mit Startdatum, neueste zuerst
Einheit im Detail	/verlauf/einheit/:id	Übungen in Reihenfolge mit ihren Sätzen, Notiz, Löschen.
Konto	/konto	Vor- und Nachname, E-Mail und Passwort ändern (Dialoge mit Passwortbestätigung), Abmelden, Konto löschen

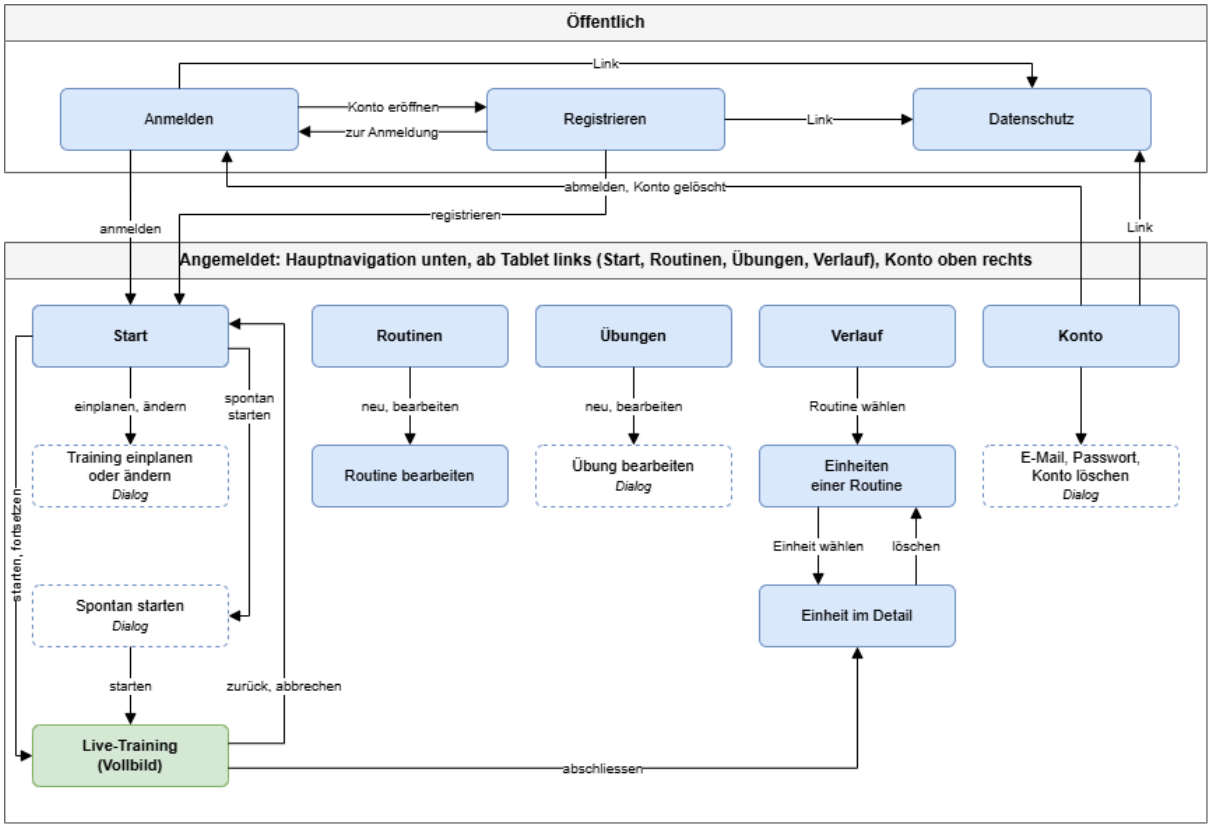


Abbildung 25: Navigation zwischen den Screens

6.5.3 Wichtige Elemente

Es werden überall die gleichen Gestaltungsgrundsätze verwendet. Die Standardkomponenten kommen aus Material UI. Folgende Elemente werden an verschiedensten Stellen eingesetzt.

Tabelle 85: Wichtige Elemente

Element	Einsatz
Untere Navigation, Seitenleiste	Wechsel zwischen den vier Bereichen Wird immer angezeigt, ausser bei einem Dialog oder Livetraining.
Dialog	Kurze Formulare als Popup und jede Bestätigung vor dem Löschen
Snackbar und Alert	Bestätigt Erfolg kurz als Popup, das nach wenigen Sekunden verschwindet. Fehler zeigen einen deutschen Text je nach Statuscode.
Feldfehler	React Hook Form zeigt Fehler direkt beim Feld, die Eingabe bleibt erhalten.
Satz-Eingabe	Drei Zahlenfelder mit numerischer Tastatur. Komma oder Punkt sind erlaubt. Beim Fokus wird der Wert markiert, damit eine Eingabe ihn ersetzt.
Schaltflächen	Mindestens 44 × 44 Pixel, sodass die Touchfähigkeit gegeben ist.
Formate	Deutsche Texte Datum wird mit den Wochentagen angezeigt: Di., 20.10.2026 Gewicht in kg

7 Entwicklung

7.1 Setup & Infrastruktur (Lokal)

Als Entwicklungsumgebung für das Backend wird IntelliJ IDEA 2026.2.0.1 verwendet.

Für das Frontend wird WebStorm 2026.2.3 verwendet.

Lokal ist auf einem Windows 11 Computer ein Docker Desktop installiert, um die Docker Images zu testen.

Das GitHub Repository https://GitHub.com/bernetlennard/digital-training-platform wurde erstellt.

Ein Git Workflow startet die Maven Tests im Backend und im Frontend Lint, Typprüfung, Vitest und Build und ist nur erfolgreich, wenn alle durchlaufen. Mit einer Regel im Repository kann man festlegen, dass Branches nur auf main gemerged werden dürfen, wenn ein Pull Request gemacht wurde und dieser Workflow erfolgreich durchläuft beim Commit. Somit stellt man sicher, dass in der Produktion nicht aus Versehen ein kaputter Stand deployed wird. Entwickelt wird also auf einem neuen Branch «develop».

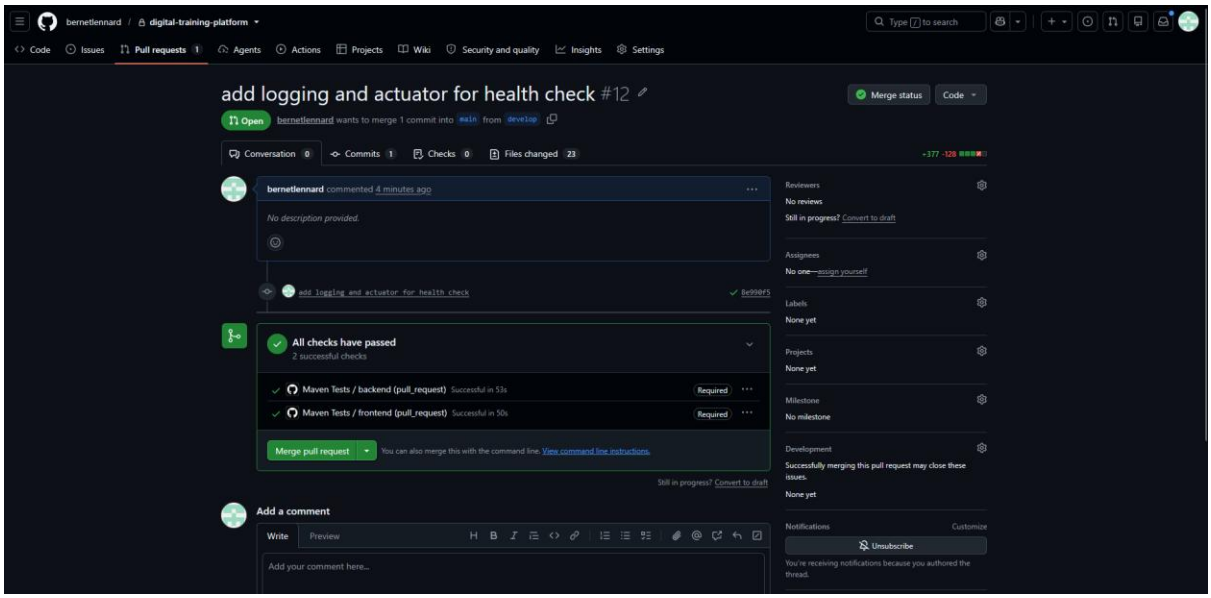


Abbildung 26: Pull Request mit erfolgreichen Tests

7.2 Backend

7.2.1 Spring Boot

In der Entscheidung E5 wurde Spring Boot gewählt. Spring Boot baut auf Java Code auf. Java läuft nicht direkt auf dem Prozessor, sondern in der Java Virtual Machine. Die JVM ist plattformunabhängig und kann lokal auf Windows, im Container auf Linux oder anderen Plattformen laufen. Während dem Entwickeln und Kompilieren braucht man nur ein sogenanntes Java Development Kit (JDK).

Klassischerweise werden Java Applikationen als WAR (komprimiertes Dateiarchiv im ZIP-Format, das eine vollständige Java-Webanwendung bündelt) auf einen Applikationsserver wie

WildFly oder GlassFish deployt. Den Server muss man separat installieren und betreiben. Mit Spring Boot steckt ein Tomcat-Webserver in der Applikation und man braucht nur ein einziges JAR (Java-Archive).

Beim Start einer Spring-Boot-Applikation wird main in TrainingplatformApplication aufgerufen, die SpringApplication.run aufruft. Anschliessend werden die Konfigurationen geladen wie application.properties, das .env File oder Umgebungsvariablen. Spring Boot scannt dann die Files und findet die Controller, Services, Repositories und Konfigurationen. Daraus entstehen sogenannte Beans welche miteinander verdrahtet werden. [17]

Die Auto-Configuration von Spring Boot richtet Verbindungspool, Hibernate, JSON-Umwandlung und Security-Filterkette ein. Flyway prüft dann, ob die Datenbank noch migriert werden muss. Anschliessend wird auch nochmals von Hibernate mit validate geprüft, ob das Datenbankschema den Entitäten im Code entspricht. Dann startet der angesprochene Tomcat-Webserver auf dem Port 8080 und der Actuator meldet «UP». [18]

7.2.2 Vorgehen nach TDD

Die Entwicklung erfolgt nach Test Driven Development (TDD). Dies wurde schon ganz am Anfang des Projekts als Massnahme festgelegt. Jede Funktion entsteht aus einem vorher festgelegten Testfall.

1. **Grobstruktur aus den Klassendiagrammen:** Alle Klassen wurden aus dem bestehenden Klassendiagramm übernommen und die nötigen Attribute und Methodensignaturen wurden festgelegt.
2. **Definitive Testfälle:** Mit dieser Grobstruktur können die definierten Testfälle angelegt werden.
3. **Red, Green, Refactor:** Als erstes wird der Test geschrieben (rot). Mit möglichst wenig Code den Testfall erfüllen (grün). Anschliessend prüfen, ob man im Code irgendwie aufräumen kann (refactor).

Umgesetzt wird von innen nach aussen (Entität, Service, Controller) in der Reihenfolge der Abhängigkeiten: Benutzerverwaltung, Übungen, Routinen, Planung, Live-Protokollierung, Historie.

Es kann auch vorkommen, dass während der Entwicklung festgestellt wird, dass man einen Testfall nicht korrekt erfasst hat, oder dass etwas geändert wird. Die Testfälle sind also während der Entwicklung nicht in Stein gemeisselt, sondern werden, wo nötig fortlaufend angepasst.

7.2.3 Testfälle

- Unit-Tests mit JUnit, Mockito und AssertJ prüfen Entitäten und Services. Controller, Validierung und Security werden mit @WebMvcTest getestet. Die Uhr ist fest eingestellt, damit Datumsregeln reproduzierbar sind.
- Ein Testfall beschreibt eine Geschäftsregel und wird mit einer oder mehreren Testmethoden über alle betroffenen Schichten umgesetzt. Jede Methode trägt die ID des Testfalls im @DisplayName
- Manuelle Systemtests laufen über HTTP-Anfragen gegen das lokale Backend mit PostgreSQL. Sie prüfen, was nur mit der echten Datenbank geht (Kaskaden, Constraints, Abfragen).

- Die Unit-Tests laufen lokal und bei jedem Pull Request in der Pipeline. Ein roter Test blockiert Merge und Deployment.

7.2.3.1 Unit-Tests Backend

Tabelle 86: Unit-Tests Backend

ID	Testfall	Erwartetes Ergebnis
Querschnitt		
UT-01	Ungültige Eingaben, fachliche Fehler und unerwartete Fehler	400 mit Feldfehlern, 401/404/409/422/429 bzw. 500 im einheitlichen Fehlerformat, ohne Stacktrace und interne Details
UT-02	Zugriff ohne, mit abgelaufenem oder falsch signiertem Token; Zugriff auf fremde oder gelöschte Datensätze	401 (ausser Registrierung, Login, Health-Check) 404 bei fremden Daten Die Benutzer-ID kommt immer aus dem Token
FT1 Benutzerverwaltung		
UT-03	Registrieren und anmelden	E-Mail getrimmt, klein geschrieben und eindeutig (409) Passwort 15–64 Zeichen (400); Argon2id-Hash Cookie HttpOnly, Secure, SameSite=Strict, 8 Stunden
UT-04	Falsches Passwort, unbekannte E-Mail, 5 Fehlversuche in Folge	Identische 401-Antwort; danach 15 Minuten gesperrt (429), auch für E-Mails ohne Konto
UT-05	Profil ändern, E-Mail und Passwort ändern, Konto löschen, abmelden	Änderungen gespeichert E-Mail, Passwort und Löschung nur mit korrektem Passwort (401) Löschung in einer Transaktion, danach wird das alte Token abgewiesen

FT2 Übungskatalog		
UT-06	Übung anlegen und ändern	Name und Muskelgruppe Pflicht (400) Name je Sportler eindeutig ohne Gross-/Kleinschreibung (409), gelöschte Übungen zählen nicht
UT-07	Übung löschen	Soft gelöscht In einer nicht gelöschten Routine verwendet ergibt 409
FT3 Trainingsroutinen		
UT-08	Routine anlegen und ändern	Mindestens eine Übung (400), keine Übung doppelt (422), Positionen 1..n in der Reihenfolge des Requests, Name eindeutig (409)
UT-09	Routine löschen bzw. ändern nach dem Start einer Einheit	Soft gelöscht, geplante Einheiten hart gelöscht Gestartete Einheiten behalten ihre kopierten Übungen
FT4 Trainingsplanung		
UT-10	Einheit einplanen, ändern und entfernen	Datum ab heute nach Europe/Zurich (sonst 422) Überfällige Einheiten nur noch löschbar Nur Einheiten mit Status «geplant» änderbar (sonst 404)
FT5 Live-Protokollierung		
UT-11	Geplante Einheit oder Routine spontan starten	Status «aktiv», Übungen der Routine in Reihenfolge kopiert Zweite aktive Einheit ergibt 409
UT-12	Sätze erfassen, ändern, verschieben und löschen	Satznummern immer lückenlos ab 1 Gewicht 0–1000 kg mit max. 2 Nachkommastellen, 1–100 Wiederholungen, Pause 0–3600 s (400) Zielposition ausserhalb (422)
UT-13	Vorschlag für den nächsten Satz	Sätze mit gleicher Einheit und Routine aus der letzten Trainingseinheit als Response laden. Persistieren erst durch Request.

UT-14	Einheit abschliessen bzw. abbrechen	Abschluss nur mit mindestens einem Satz (409), Übungen ohne Satz entfernt Abbruch setzt geplante Einheit zurück, spontane wird gelöscht
FT6 Trainingshistorie		
UT-15	Historie lesen und Einheit löschen	nur abgeschlossene Einheiten, neueste zuerst, auch für gelöschte Routinen; Einheit wird soft gelöscht; abgeschlossene Einheiten sind nicht mehr bearbeitbar (404)

7.2.3.2 Manuelle Systemtests Backend

Für die manuelle Systemtests im Backend können direkt aus der IDE HTTP-Requests abgesetzt werden. Somit wird die lokale Datenbank angesprochen. In IntelliJ gibt es die Möglichkeit, die Response eines REST-Calls direkt zu prüfen. [19]. Diese JavaScript-Prüfungen wurden mit Unterstützung von Claude Opus 5.5 erstellt.

Tabelle 87: Manuelle Systemtests Backend

ID	Testfall	Erwartetes Ergebnis
ST-01	Registrieren, anmelden, abmelden, Konto mit Trainingsdaten löschen	Alle Schritte erfolgreich Vom Sportler sind keine Daten mehr in der Datenbank vorhanden.
ST-02	Übung und Routine anlegen Training spontan starten Sätze erfassen Sätze verschieben und löschen Training abschliessen Historie abrufen	Die Einheit erscheint korrekt in der Historie mit allen Übungen und Sätzen, die dazugehören.
ST-03	Zweiter Sportler greift auf Daten des ersten zu	Liefert jeweils ein 404-Fehler. Es werden keine Daten gelesen oder verändert.

7.2.4 Herausforderungen und Lösungen

Während der Entwicklung lief nicht immer alles perfekt nach Plan. Es wurden verschiedenste Herausforderungen überwunden, bei denen es gute Lösungen brauchte. Ein zentraler Punkt war die erste Version, wie die vorgeschlagenen Sätze ermittelt werden. Die erste Version hat sich in der Bedienung nicht richtig angefühlt. Es wurde der letzte Satz der vorherigen Trainingseinheit übernommen und wenn man einen Satz mehr erfasste, wurden die Werte der aktuellen Trainingseinheit vorgefüllt. Schlussendlich gab es eine kleine Anpassung an den Anforderungen, Response DTOs und natürlich die Business Logik dafür im Service. Die Datenbank konnte man so lassen, wie sie war.

Ein anderes Problem, dass ich erst hatte, war ein abgelaufenes Cookie, das den Login blockierte. Der Browser schickt das alte access_token auch an /auth/login. Dieses wird direkt abgelehnt, bevor der Login, der das neue generiert, überhaupt läuft. Als Lösung wurde im CookieBearerTokenResolver das Cookie ignoriert auf register, login, logout und health.

Wenn ein neuer Satz erstellt wird oder ein Training startet, muss man auch an die neu gespeicherte ID kommen. Dafür wurde jeweils auf dem Repository die Methode .flush gebraucht.

Um die zirkulären Abhängigkeiten zwischen den Fachbereichen aufzulösen, wurden Interfaces benutzt. Die Übung muss die Routinen kennen, die Routine das Training und das Training wiederum Routinen und Übungen. Als Lösung kommen die Interfaces ExerciseUsageChecker und RoutineDeletionListener zum Zug.

Ein bekanntes Problem ist jeweils, dass die Software die korrekte Zeitzone erkennt. Wenn jetzt der Server nicht die gleiche Zeit hat wie die Schweizer Zeit, kann es zu Fehlern beim Datum kommen. Als Lösung wurde dafür eine zentrale Clock eingeführt mit der Zeit von Europe/Zurich.

7.2.5 README

Im Repository ist ein README.md File hinterlegt. Dies dient als Anleitung für das Setup. Das Ziel ist es, dass Drittpersonen der Anleitung in diesem README folgen können und danach die Applikation mit der Datenbank lokal läuft.

Die nächsten Kapitel widerspiegeln den Inhalt des README Files.

7.2.6 Voraussetzungen

Für den lokalen Betrieb muss folgende Software installiert sein:

- **JDK 25:** Die Umgebungsvariable `JAVA_HOME` muss auf das JDK zeigen, da der Maven Wrapper sie benötigt.
- **PostgreSQL 18:** Lokal laufender Datenbankserver.
- **Git:** Zum Klonen des Repositorys.

Maven muss nicht installiert sein. Der Maven Wrapper (mvnw bzw. unter Windows mvnw.cmd) lädt die passende Version automatisch herunter.

7.2.7 Umgebungsvariablen

In src/main/resources/application.properties stehen nur Standardwerte. Diese werden über Umgebungsvariablen bzw. über ein .env-File im /backend-Verzeichnis überschrieben. Die Properties müssen also nicht angepasst werden, es reicht, die Variablen zu setzen:

Tabelle 88: Umgebungsvariablen

Variable	Pflicht	Standardwert	Beschreibung
DB_PASSWORD	Ja	–	Passwort des Datenbankbenutzers
JWT_SECRET	Ja	–	Schlüssel zum Signieren der JWT, mind. 32 Zeichen
DB_URL	Nein	jdbc:postgresql://localhost:5432/digital_training_platform	JDBC-URL der Datenbank
DB_USERNAME	Nein	postgres	Datenbankbenutzer

Fehlt eine Pflichtvariable oder ist das JWT_SECRET kürzer als 32 Zeichen, startet die Anwendung nicht.

Beispiel für ein `.env`-File:

```
DB_PASSWORD=mein-db-passwort
JWT_SECRET=ein-zufaelliger-schluessel-mit-mindestens-32-zeichen
```

7.2.8 Verwendete Technologien

Dieses Backend wurde mit folgenden Technologien entwickelt:

- **Java:** Version 25
- **PostgreSQL:** Version 18
- **Maven:** Version 3.9.16 (über den Maven Wrapper)
- **Spring Boot:** Version 4.1.1

Die meisten Abhängigkeiten werden ohne Versionsangabe im pom.xml eingebunden. Ihre Versionen gibt der spring-boot-starter-parent vor. Dieser erbt von der BOM (Bill of Materials) spring-boot-dependencies, welche für alle unterstützten Bibliotheken eine getestete und zueinander passende Version festlegt. Die folgenden Versionen werden durch Spring Boot 4.1.1 verwaltet:

- **Spring Framework:** Version 7.0.9
- **Spring Security:** Version 7.1.1
- **Spring Data JPA:** Release Train 2026.0.1
- **Hibernate:** Version 7.4.5.Final
- **Flyway:** Version 12.4.0
- **PostgreSQL JDBC-Treiber:** Version 42.7.13
- **JUnit Jupiter:** Version 6.0.3
- **Mockito:** Version 5.23.0
- **Lombok:** Version 1.18.46

Bibliotheken, welche die BOM nicht kennt, werden mit einer eigenen Version eingebunden:

- **springdoc-openapi:** Version 3.1.0
- **Bouncy Castle:** Version 1.86 (wird für das Argon2-Passwort-Hashing benötigt)

Die tatsächlich aufgelösten Versionen lassen sich mit ./mvnw dependency:tree anzeigen.

7.2.9 Lokales Setup & Installation

7.2.9.1 Schritt 1: Repository klonen

Klonen Sie dieses Repository auf Ihren lokalen Computer:

```
git clone https://GitHub.com/bernetlennard/digital-training-platform.git
cd digital-training-platform/backend
```

7.2.9.2 Schritt 2: Datenbank-Setup

Bevor die Anwendung gestartet werden kann, muss die PostgreSQL-Datenbank lokal eingerichtet werden.

Schritt-für-Schritt-Einrichtung:

1. Stellen Sie sicher, dass der PostgreSQL-Server läuft.
2. Erstellen Sie die Datenbank digital_training_platform auf dem PostgreSQL-Server.
3. Setzen Sie die Umgebungsvariablen gemäss Kapitel «Umgebungsvariablen» im `.env`-File.
4. Dank der Flyway-Migration werden beim ersten erfolgreichen Start der Applikation alle Tabellen erstellt. Die Datenbank muss also leer sein.

7.2.9.3 Schritt 3: Anwendung starten

Die Anwendung kann in der lokalen Umgebung auf zwei Arten gestartet werden:

Option A: Über die IDE (z.B. IntelliJ IDEA)

Führen Sie die Hauptklasse src/main/java/ch/bernet/trainingplatform/TrainingplatformApplication.java aus.

Option B: Über Maven (Kommandozeile)

Navigieren Sie im Terminal ins `/backend`-Verzeichnis des Projekts und führen Sie folgenden Befehl aus:

```
./mvnw spring-boot:run
```

7.2.10 Technische Details

7.2.10.1 Konfiguration

Die Konfigurationen werden nicht hardcodiert im Code hinterlegt. Dafür gibt es in Spring Boot ein sogenanntes Property File. Darin stehen Merkmale wie die Datenbank-URL, JPA-Konfigurationen und das JWT-Secret. Es gibt zusätzlich zu diesem File lokal noch ein .env-File. Hier werden Passwörter und Secrets abgelegt, welche nicht in das GitHub-Repository gepusht werden. Das Property File verweist einfach auf diese Secrets.

7.2.10.2 Dockerfile

Für das Backend wurde ein Dockerfile erstellt. Im Dockerfile wird Swagger ausgeschaltet, indem die Properties SPRINGDOC_APIDOCS_ENABLED und SPRINGDOC_SWAGGERUI_ENABLED auf false gesetzt werden.

7.2.10.3 Flyway-Migration

Beim Start der Spring-Boot-App schaut Flyway im Ordner «db/migration» nach SQL-Dateien. Sie müssen dabei nach dem Schema «V1__...», «V2__...» benannt sein. Flyway führt jede Datei genau einmal aus, in der Reihenfolge der Versionsnummer. Sobald eine Datei gelaufen ist, wird dies in der Tabelle «flyway_schema_history» gemerkt. Anschliessend prüft Hibernate mit «ddl-auto=validate», ob die Entitäten zum Schema passen. [20]

7.2.10.4 Spring Security

Mithilfe von Spring Security wird für die Benutzerauthentifizierung ein JWT erzeugt. Dieses wird beim Login als Cookie access_token (HttpOnly, Secure, SameSite=Strict) gesetzt. Das JWT wird nur aus diesem Cookie gelesen, nicht aus dem Authorization-Header. Das Token ist standardmässig 8 Stunden gültig (konfigurierbar über app.jwt.validity).

Die Anwendung ist stateless, es werden also keine Server-Sessions angelegt. Jeder Request wird allein anhand des Tokens authentifiziert. Der CSRF-Schutz ist deaktiviert, da das Cookie mit SameSite=Strict bei Requests von fremden Seiten nicht mitgeschickt wird.

Passwörter werden nie im Klartext gespeichert, sondern mit Argon2id gehasht.

Folgende Endpunkte sind ohne Token erreichbar:

Tabelle 89: Öffentliche Endpunkte

Methode	Pfad
POST	/api/v1/auth/register
POST	/api/v1/auth/login
POST	/api/v1/auth/logout
GET	/api/actuator/health
*	/error
*	/swagger-ui.html, /swagger-ui/**, /v3/api-docs/**

Alle anderen Endpunkte brauchen ein gültiges JWT, ansonsten wird der Request mit 401 abgelehnt:

Tabelle 90: Geschützte Endpunkte

Methode	Pfad
Benutzer	
GET	/api/v1/users/me
PUT	/api/v1/users/me
PUT	/api/v1/users/me/email
PUT	/api/v1/users/me/password
DELETE	/api/v1/users/me
Übungen	
GET	/api/v1/exercises
GET	/api/v1/exercises/muscle-groups
POST	/api/v1/exercises
PUT	/api/v1/exercises/{exerciseId}
DELETE	/api/v1/exercises/{exerciseId}
Routinen	
GET	/api/v1/routines
GET	/api/v1/routines/{routineId}
POST	/api/v1/routines
PUT	/api/v1/routines/{routineId}
DELETE	/api/v1/routines/{routineId}
Trainingsplanung	
GET	/api/v1/sessions/planned
POST	/api/v1/sessions/planned
PUT	/api/v1/sessions/planned/{sessionId}
DELETE	/api/v1/sessions/planned/{sessionId}
Live-Training	
POST	/api/v1/sessions/planned/{sessionId}/start
POST	/api/v1/sessions
GET	/api/v1/sessions/active
PUT	/api/v1/sessions/active/note

POST	/api/v1/sessions/active/exercises/{sessionExerciseId}/sets
PUT	/api/v1/sessions/active/sets/{setId}
PUT	/api/v1/sessions/active/sets/{setId}/position
DELETE	/api/v1/sessions/active/sets/{setId}
POST	/api/v1/sessions/active/finish
POST	/api/v1/sessions/active/cancel
Historie	
GET	/api/v1/history/routines
GET	/api/v1/history/routines/{routineId}/sessions
GET	/api/v1/history/sessions/{sessionId}
DELETE	/api/v1/history/sessions/{sessionId}

7.2.10.5 Transaction Handling

Jegliche Service-Methoden enthalten jeweils die Annotation @Transactional, sobald eine Transaktion auf der Datenbank ausgeführt werden muss. Über die Propagation kann man steuern, ob eine bestehende Transaktion übernommen wird, oder ob zwingend eine neue Transaktion erstellt werden soll. Nach dem Ablauf der Methode wird die Transaktion auf der Datenbank committed.

7.2.10.6 Exception Handling

Jegliche Exceptions werden im Format des ErrorResponse-Records zurückgegeben. Somit lassen sich übersichtliche und einheitliche Fehlermeldungen bilden. Dies stellt der GlobalExceptionHandler sicher. Jegliche Messages, welche den Exceptions übergeben werden, sind als Konstanten definiert. Somit lassen sich einheitliche Fehlermeldungen an einem gebündelten Ort im Code sauber verwalten.

7.2.10.7 Swagger

Swagger ist in der Anwendung integriert und liefert eine OpenAPI-Dokumentation. Es wird OpenAPI 3.1 verwendet. In diesem Projekt wird jeder Controller mit einem Tag beschriftet. Über die Klasse ApiStatus werden pro Endpunkt die HTTP-Statuscodes beschrieben. Diese Dokumentation hilft dabei, die REST-API zu entwerfen, zu bauen, zu dokumentieren, zu testen und zu nutzen.

Swagger ist nur lokal aktiv, im Docker-Container ist es ausgeschaltet (siehe Dockerfile). Nach dem Start ist es erreichbar unter:

- **Swagger UI:** http://localhost:8080/swagger-ui.html
- **OpenAPI-Spezifikation:** http://localhost:8080/v3/api-docs

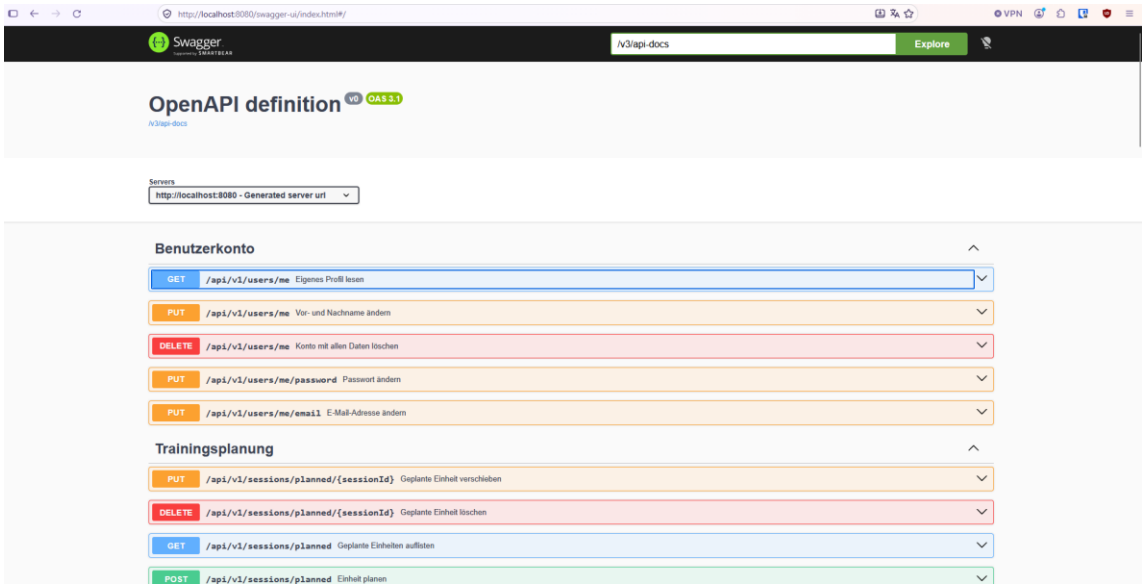


Abbildung 27: Swagger UI

Die Swagger-Pfade sind in Spring Security freigegeben. Im Container liefern sie trotzdem 404, da Swagger dort ausgeschaltet ist.

Da das JWT nur aus dem Cookie gelesen wird, nützt der «Authorize»-Button von Swagger nichts. Stattdessen in Swagger zuerst POST /api/v1/auth/login ausführen. Danach schickt der Browser das Cookie bei jedem Request automatisch mit.

7.2.10.8 API-Versionierung

Alle Endpunkte haben die Version im Pfad, z.B. /api/v1/users/me.

Es wird die eingebaute API-Versionierung von Spring Framework 7 verwendet. Die Klasse ApiVersionConfig schaltet sie ein. Die Versionierung gilt nur für Pfade unter /api/. Swagger und /error haben keine Version. Jeder Controller hat im @RequestMapping den Pfad /api/v{version}/... und die Version 1, z.B. @RequestMapping(path = "/api/v{version}/users/me", version = "1").

Somit erkennt man direkt am Pfad, welche Version die Schnittstelle verwendet. Wenn man einen Endpunkt ändert, bricht man den Schnittstellenvertrag und das bestehende Frontend funktioniert nicht mehr. Daher kommt die Änderung in eine neue Version v2. Die Version v1 bleibt daneben bestehen. So kann man schrittweise umsteigen. Eine Version, die es nicht gibt (z.B. /api/v2/...), wird mit 400 abgelehnt. Ein Pfad ohne Version (z.B. /api/users/me) ergibt 404.

7.2.10.9 Actuator

Mit dem Spring Boot Actuator erhält man die Möglichkeiten, die die Applikation einfach zu überwachen. Alles, was man dafür tun muss, ist die Dependency hinzuzufügen.

Hier wird er vor allem benutzt, um während dem Deployment zu prüfen, ob der Status Healthy ist, also die Springboot Applikation sauber hochgefahren wurde. Der GET-Endpunkt /api/actuator/health ist ohne Token erreichbar. Dies ist so in der SecurityConfig festgelegt.

Die Antwort ist {"status":"UP"} mit 200, wenn alles in Ordnung ist. Ansonsten kriegt man den Status DOWN mit 503. [21]

7.3 Frontend

Für die Entwicklung im Frontend hat man als Ausgangslage die Anforderungen, REST-Schnittstelle und das GUI-Konzept aus der Konzeption. Mit diesen Inputs ist schon sehr viel klar, wie es entwickelt werden muss.

Durch die begrenzten Vorkenntnisse und den groben Verzug im Zeitplan wurde die Entwicklung stark unterstützt von Claude Code (Opus 5.5). In der aktuellen Zeit wird es immer üblicher, mit der Unterstützung von KI zu programmieren. Mit diesem Ansatz der Programmierung ist es aber umso wichtiger, saubere Code Reviews und manuelle Tests durchzuführen. Ansonsten verliert man sehr schnell die Übersicht des Codes und es können sich ungewollte Bugs einschleichen.

Darum sind für das Frontend vor allem die manuelle Systemtests wichtig. Sie sollen das gesamte GUI-Konzept abdecken.

7.3.1 Manuelle Systemtests

Ähnlich wie beim Backend wird hier auch ein TDD-Ansatz verfolgt. Anstatt von Anfang an feste Tests im Code zu erstellen, werden die manuelle Systemtests festgelegt. Im iterativen Prozess wird der benötigte Code pro Systemtest generiert, akribisch geprüft und sauber getestet. Dies wird gemacht, bis alle vorher definierten Manuelle Systemtests erfolgreich sind.

Die Testfälle bauen aufeinander auf und sollten in dieser Reihenfolge lokal ausgeführt werden.

Tabelle 91: Manuelle Systemtests Frontend

ID	Testfall	Erwartetes Ergebnis
FT1 Benutzerverwaltung		
STF-01	«Registrieren» → Passwort unter 15 Zeichen → «Konto erstellen»	Fehler beim Passwortfeld, Eingaben bleiben
STF-02	E-Mail, gültiges Passwort, Vor- und Nachname → «Konto erstellen»	Startseite Name unter «Konto» sichtbar
STF-03	«Konto» → «Abmelden»	Anmeldeseite
STF-04	«Registrieren» mit derselben E-Mail	Meldung «Konto besteht bereits», Eingaben bleiben
STF-05	Ohne Anmeldung Adresse /routinen aufrufen	Weiterleitung auf Anmelden
STF-06	«Datenschutzerklärung» → «Zurück»	Datenschutz ohne Anmeldung lesbar
STF-07	Falsches Passwort → «Anmelden»	Fehlermeldung, E-Mail bleibt stehen
STF-08	Unbekannte E-Mail: 6× «Anmelden»	Ab dem 6. Versuch Meldung «15 Minuten gesperrt»
STF-09	Richtiges Passwort → «Anmelden»	Angemeldet, Hauptnavigation sichtbar
STF-10	«Konto» → Vor- und Nachname ändern → «Speichern»	Meldung «Profil gespeichert»
STF-11	«E-Mail-Adresse ändern» mit falschem Passwort	Fehler im Dialog, Dialog bleibt offen
STF-12	«E-Mail-Adresse ändern» mit richtigem Passwort	Neue E-Mail angezeigt
STF-13	«Passwort ändern» → abmelden → mit neuem Passwort anmelden	Anmeldung klappt

STF-14	Abmelden, zweites Konto registrieren, eine Übung anlegen «Konto» → «Konto löschen» → Passwort → «Endgültig löschen»	Anmeldeseite Anmeldung mit dem zweiten Konto schlägt fehl
FT2 Übungskatalog		
STF-15	«Übungen» → «Neue Übung» → leer «Speichern»	Fehler bei Namen und Muskelgruppe
STF-16	Name, Muskelgruppe, lange Beschreibung → «Speichern»	Übung mit Muskelgruppe in der Liste Beschreibung auf 2 Zeilen gekürzt
STF-17	Zwei weitere Übungen anlegen	Liste alphabetisch
STF-18	«Neue Übung» mit vorhandenem Namen in anderer Schreibweise	Meldung «Name vergeben», Dialog bleibt offen
STF-19	Übung antippen → Muskelgruppe ändern → «Speichern»	Änderung in der Liste
STF-20	Übung antippen → «Löschen» → bestätigen	Übung verschwindet aus der Liste
FT3 Trainingsroutinen		
STF-21	«Routinen» → «Neue Routine» → Name → «Speichern»	Hinweis, dass eine Übung fehlt
STF-22	«Übungen hinzufügen» → zwei Übungen antippen → «2 hinzufügen» → «Speichern»	Übungen in Tipp-Reihenfolge Routine mit «2 Übungen» in der Liste
STF-23	Routine öffnen → Pfeil nach oben → «Speichern» → wieder öffnen	Neue Reihenfolge gespeichert
STF-24	«Neue Routine» mit vorhandenem Namen → «Speichern»	Meldung «Name vergeben», Eingaben bleiben
STF-25	«Übungen» → Übung aus der Routine → «Löschen» → bestätigen	Fehlermeldung im Dialog, Übung bleibt
STF-26	Zweite Routine anlegen und auf «Start» einplanen Zweite Routine öffnen → «Löschen» → bestätigen	Rückfrage weist auf geplante Trainings hin Routine und geplantes Training verschwinden
FT4 Trainingsplanung		
STF-27	«Start» → «Einplanen» → Routine, Datum heute → «Einplanen»	Training unter «Geplante Trainings» mit Wochentag und «heute»
STF-28	Zweites Training in einer Woche einplanen	Nach Datum sortiert, nur das nächste mit gefülltem «Starten»
STF-29	Im Dialog ein Datum vor heute wählen	Nicht möglich bzw. Fehler beim Datum
STF-30	Zweites Training: Stift-Symbol → Datum ändern → «Speichern»	Neues Datum angezeigt
STF-31	Zweites Training: Papierkorb-Symbol → bestätigen	Training verschwindet
FT5 Live-Protokollierung		
STF-32	Geplantes Training → «Starten»	Vollbild ohne Navigation, Startzeit und Dauer oben Übungen in der Reihenfolge der Routine
STF-33	Ohne Satz «Abschliessen» → bestätigen	Hinweis im Dialog, zuerst einen Satz zu erfassen
STF-34	Gewicht leer → «Satz 1 speichern»	Fehler beim Gewicht

STF-35	Gewicht mit Komma, Wdh., Pause → «Satz 1 speichern» Satz 2 und 3 erfassen	Sätze 1–3 in der Liste, Fortschritt nachgeführt
STF-36	Satz antippen → Werte ändern → «Speichern»	Neue Werte angezeigt
STF-37	Satz 3 antippen → Position «Satz 1» → «Speichern»	Satz steht zuoberst, Nummern 1–3
STF-38	Satz antippen → «Löschen» → bestätigen	Sätze lückenlos ab 1 nummeriert
STF-39	Seite neu laden	Training mit allen Sätzen wieder offen
STF-40	Zurück-Pfeil oben links	Start mit dem Training zuoberst «Starten» und «Spontan starten» gesperrt
STF-41	«Fortsetzen»	Zurück im Training
STF-42	Notiz eingeben, Feld verlassen	Hinweis «Gespeichert»
STF-43	«Abschliessen» → bestätigen	Detail des Trainings im Verlauf
STF-44	Gleiche Routine: «Spontan starten» → Routine antippen	Eingabe mit Satz 1 des letzten Trainings vorbelegt Übrige Sätze als «Satz n · offen»
STF-45	Vorgeschlagene Sätze speichern «Abschliessen» → bestätigen	Häkchen, sobald alle Sätze erfasst sind Zweites Training im Verlauf
STF-46	«Spontan starten» → Routine → «Abbrechen» → bestätigen	Training verschwindet, Startseite
STF-47	Training einplanen → «Starten» → «Abbrechen» → bestätigen	Training wieder unter «Geplante Trainings»
FT6 Trainingshistorie		
STF-48	«Verlauf»	Routine mit abgeschlossenen Trainings
STF-49	Routine antippen	Zwei Trainings, neueste zuerst
STF-50	Training antippen	Übungen mit Sätzen und Notiz
STF-51	«Training löschen» → bestätigen	Training nicht mehr im Verlauf
STF-52	Routine unter «Routinen» löschen → «Verlauf»	Routine weiter sichtbar, als «gelöscht» markiert

7.3.1.1 Status Trainingseinheit

Die folgenden manuelle Systemtests gehen alle Schritte des Status einer Trainingseinheit gemäss dem Zustandsdiagramm einmal im Frontend durch. Diese Tests ergänzen die manuelle Systemtests und werden lokal in dieser Reihenfolge ausgeführt. Als Voraussetzung braucht man ein Konto ohne geplante Trainings mit den Routinen A und B, die je zwei Übungen haben.

Tabelle 92: Manuelle Tests Status Trainingseinheit

ID	Testfall	Erwartetes Ergebnis
Geplant		
STS-01	«Start» → «Einplanen» → Routine A, Datum morgen → «Einplanen»	Training unter «Geplante Trainings» mit «nächstes»
STS-02	Stift-Symbol → Datum heute → «Speichern»	Markierung «heute»
STS-03	In der Datenbank das Datum auf gestern setzen → Seite neu laden	Markierung «vergangen» Kein Stift-Symbol mehr, nur «Starten» und Papierkorb

STS-04	Papierkorb → «Löschen»	Training verschwindet aus der Planung
Aktiv		
STS-05	Routine A einplanen → «Starten»	Vollbild mit Startzeit und Dauer
STS-06	Pfeil oben links	«Training läuft» zuoberst, nicht mehr unter «Geplante Trainings» «Starten» und «Spontan starten» gesperrt
STS-07	«Fortsetzen» → einen Satz erfassen → «Abbrechen» → «Training abbrechen»	Training wieder unter «Geplante Trainings»
STS-08	«Spontan starten» → Routine A → «Abbrechen» → «Training abbrechen»	Training weder geplant noch im Verlauf
Abgeschlossen		
STS-09	Geplantes Training «Starten» → ohne Satz «Abschliessen» → «Abschliessen»	Hinweis im Dialog, Training bleibt aktiv
STS-10	Nur bei der ersten Übung einen Satz erfassen → «Abschliessen» → «Abschliessen»	Detail im Verlauf nur mit der ersten Übung Nicht mehr unter «Geplante Trainings»
STS-11	Adresse /training aufrufen	«Kein Training aktiv», das Training lässt sich nicht mehr ändern
STS-12	Training im Verlauf öffnen → «Training löschen» → «Löschen»	Training nicht mehr im Verlauf
Routine gelöscht		
STS-13	Routine B zweimal einplanen und eines starten → Pfeil oben links → Routine B löschen	Das andere geplante Training ist weg «Training läuft» bleibt
STS-14	«Fortsetzen» → «Abbrechen» → «Training abbrechen»	Training ist nicht wieder eingeplant, sondern ganz gelöscht

7.3.1.2 Löschkonzept

Diese manuellen Testfälle decken das Löschkonzept aus dem Datenschutz und dem Datenmodell vollständig ab. Hier wird nicht nur geprüft, ob ein Eintrag jeweils auch im Frontend verschwindet, sondern auch, ob er in der Datenbank wirklich hart bzw. soft gelöscht wird. Das Ergebnis wird jeweils direkt in der Datenbank kontrolliert.

Diese Regeln sind eigentlich auch schon im Backend mit den Unit-Tests abgesichert. Diese laufen dort aber mit Mocks und prüfen keine Fremdschlüssel, Kaskaden oder eindeutige Indizes so, wie sie im Datenmodell vorhanden sind.

Für diese Tests müssen lokal das Backend, Frontend und eine PostgreSQL laufen. Als Testdaten braucht man zwei Konten. Konto A hat drei Übungen, die Routinen A und B und ein bereits abgeschlossenes Training von Routine A. Konto B hat eine Übung, Routine und abgeschlossenes Training. Für die Prüfung des Tokens wird auch noch ein zweiter Browser verwendet.

Tabelle 93: Testfälle Löschkonzept

ID	Testfall	Erwartetes Ergebnis
Trainingseinheit		
STL-01	Training von Routine A spontan starten, drei Sätze bei der ersten Übung → Satz 2 löschen	Sätze 1 und 2 DB: Satz nicht mehr vorhanden, Satznummern 1 und 2
STL-02	Ohne Satz bei der zweiten Übung abschliessen	DB: Status abgeschlossen, zweite Übung nicht mehr vorhanden
STL-03	Geplantes Training starten, Satz erfassen → abbrechen	Wieder geplant DB: Status geplant, Startzeit leer, keine Übungen und Sätze mehr
STL-04	Spontanes Training mit einem Satz → abbrechen	DB: Einheit mit Übungen und Sätzen nicht mehr vorhanden
STL-05	Geplantes Training von Routine B starten → Routine B löschen → abbrechen	DB: Einheit nicht mehr vorhanden, nicht wieder geplant
STL-06	Geplantes Training mit dem Papierkorb löschen	DB: Einheit nicht mehr vorhanden
STL-07	Training aus STL-02 im Verlauf löschen	Nicht mehr im Verlauf DB: deleted = true, Übungen und Sätze noch vorhanden
Übung und Routine		
STL-08	Übung aus Routine A löschen	Meldung im Dialog, Übung bleibt DB: deleted = false
STL-09	Routine A einplanen → Routine A löschen	Routine und geplantes Training weg, im Verlauf als «gelöscht» DB: Routine mit deleted = true, geplante Einheit nicht mehr vorhanden
STL-10	Übung aus STL-08 nochmals löschen	Übung nicht mehr im Katalog, im Verlauf weiter mit Namen DB: deleted = true
STL-11	In der DB deleted = false setzen → «Übungen» neu laden	Übung wieder im Katalog
STL-12	Übung wieder löschen und mit gleichem Namen neu anlegen	Keine Meldung «Name vergeben»
Konto		
STL-13	Konto A zusätzlich in einem zweiten Browser anmelden Im ersten «Konto löschen» mit falschem Passwort	Fehler im Dialog, Konto bleibt DB: Fehlversuch in login_lock
STL-14	«Konto löschen» mit richtigem Passwort	Anmeldeseite Im zweiten Browser führt die nächste Aktion auf die Anmeldeseite
STL-15	Datenbank mit den notierten IDs prüfen	Keine Daten von Konto A mehr, auch nicht in login_lock Daten von Konto B unverändert
STL-16	Backend-Log nach der E-Mail von Konto A durchsuchen	Keine Treffer, nur IDs
STL-17	Mit Konto A anmelden Mit derselben E-Mail registrieren	Anmeldung schlägt fehl Registrierung klappt

7.3.2 Unit-Tests

Nach der Umsetzung des Frontends werden trotzdem noch Unit-Tests angelegt. Die meiste Logik im Frontend liegt in den Hilfsmodulen ohne React. Die Seiten und Dialoge werden über die manuelle Systemtests schon ausreichend geprüft.

Diese Unit-Tests stellen sicher, dass die Umwandlung der Eingaben, die Fehlertexte und die Formate in Zukunft nicht verändert werden, ohne dass Tests fehlschlagen.

Diese Unit-Tests wurden mit Vitest und Unterstützung von Claude Opus 5.5 erstellt.

Die Tests laufen ohne Browser und ohne Backend. Es wurde eine Uhr fest eingestellt mit einer Zeitzone. Somit laufen Tests, welche auf ein Datum und Zeitzone, prüfen in der Pipeline sauber durch.

Es werden grundsätzlich folgende drei Hilfsmodule geprüft:

- **setForm.ts**: Prüfung und Umwandlung der Satzwerte im Live-Training
- **client.ts**: Anfragen an die REST-Schnittstelle, Fehlermeldungen und IDs aus der Adresse
- **format.ts**: Formatierung für Datum, Uhrzeit, Dauer und Sätze.

Tabelle 94: Unit-Tests Frontend

ID	Testfall	Erwartetes Ergebnis
UTF-01	Satzwerte prüfen	Nur gültige Werte für Gewicht, Wiederholungen und Pause werden akzeptiert, sonst erscheint eine Fehlermeldung
UTF-02	Satz vorbelegen und umwandeln	Werte vom letzten Training sind vorbelegt, ein Gewicht mit Komma wird richtig gespeichert
UTF-03	Fehlermeldungen je Statuscode	Verständlicher deutscher Text, die Message vom Backend wird nie angezeigt
UTF-04	Requests an das Backend	Antworten und Fehler werden richtig verarbeitet, bei abgelaufener Sitzung wird man abgemeldet
UTF-05	Ungültige ID in der Adresse	Wird wie ein nicht vorhandener Eintrag behandelt (404)
UTF-06	Deutsche Formate	Datum, Zeit, Dauer und Gewicht im deutschen Format, z.B. 28.10.2026 oder 1h 05min

7.3.3 README

Genau wie im Backend wurde auch für das Frontend eine README.md Datei hinterlegt. Diese dient als Anleitung für das lokale Setup mit dem Ziel, dass Drittpersonen das Frontend zusammen mit dem Backend lokal starten können.

7.3.4 Voraussetzungen

Für den lokalen Betrieb muss folgende Software installiert sein:

- **Node.js 24.21.0:** Es muss genau diese Version sein (siehe Kapitel «Verwendete Technologien»). npm ist bei Node.js dabei.
- **Git:** Zum Klonen des Repositorys.

- **Backend:** Das Backend muss lokal laufen, sonst kann man sich nicht anmelden. Die Anleitung dazu steht im Backend-README (backend/README.md).

Das Frontend braucht keine Umgebungsvariablen.

7.3.5 Verwendete Technologien

Dieses Frontend wurde mit folgenden Technologien entwickelt:

- **Node.js:** Version 24.21.0 (nur für die Entwicklung und den Build)
- **React:** Version 19.3.0
- **TypeScript:** Version 6.0.3
- **Vite:** Version 8.3.4 (Entwicklungsserver und Build)
- **Material UI:** Version 9.5.0 (Komponentenbibliothek)
- **React Router:** Version 8.4.0 (Seiten und Navigation)
- **React Hook Form:** Version 7.90.0 (Formulare)

Für die Prüfung und Formatierung des Codes:

- **ESLint:** Version 10.12.0
- **Prettier:** Version 3.9.10
- **Vitest:** Version 5.0.4

Alle Versionen stehen fix ohne ^ im package.json. Dafür sorgt save-exact=true im .npmrc. Die Versionen aller weiteren Abhängigkeiten sind im package-lock.json festgehalten. Ebenfalls im .npmrc steht engine-strict=true. Damit bricht npm ab, wenn die installierte Node.js-Version nicht zur Angabe engines im package.json passt.

TypeScript läuft im «strict»-Modus. Dieser ist ab TypeScript 6 Standard und muss im `tsconfig` nicht extra eingeschaltet werden.

7.3.6 Lokales Setup & Installation

7.3.6.1 Schritt 1: Repository klonen

Klonen Sie dieses Repository auf Ihren lokalen Computer:

```
git clone https://GitHub.com/bernetlennard/digital-training-platform.git
cd digital-training-platform/frontend
```

7.3.6.2 Schritt 2: Abhängigkeiten installieren

Installieren Sie im /frontend-Verzeichnis die Abhängigkeiten:

```
npm ci
```

npm ci installiert genau die Versionen aus dem package-lock.json in den Ordner node_modules.

7.3.6.3 Schritt 3: Backend starten

Starten Sie zuerst das Backend gemäss Backend-README (backend/README.md). Es muss unter http://localhost:8080 laufen, da das Frontend die Requests dorthin weiterleitet (siehe Kapitel «Vite-Proxy»).

7.3.6.4 Schritt 4: Anwendung starten

Die Anwendung kann in der lokalen Umgebung auf zwei Arten gestartet werden:

Option A: Über die IDE (z.B. WebStorm)

Öffnen Sie das package.json und starten Sie das Script dev.

Option B: Über npm (Kommandozeile)

Navigieren Sie im Terminal ins /frontend-Verzeichnis des Projekts und führen Sie folgenden Befehl aus:

```
npm run dev
```

Danach ist die Anwendung erreichbar unter http://localhost:5173. Ein Konto erstellt man direkt in der Anwendung über «Registrieren». Änderungen am Code sind sofort im Browser sichtbar, ohne Neustart.

7.3.6.5 Weitere Befehle

Tabelle 95: Weitere Befehle

Befehl	Beschreibung
npm run lint	Prüft den Code mit ESLint
npm run typecheck	Prüft die Typen mit TypeScript
npm test	Führt die Tests mit Vitest aus
npm run build	Erstellt den Build für die Produktion im Ordner dist
npm run format	Formatiert den Code mit Prettier

Die ersten vier Befehle laufen auch in der GitHub-Pipeline bei jedem Pull Request auf main.

7.3.7 Technische Details

7.3.7.1 Ordnerstruktur

Der Code liegt im Ordner src:

Tabelle 96: Ordnerstruktur

Ordner	Inhalt
api/	Requests an die REST-Schnittstelle, ein File pro Bereich
auth/	Angemeldeter Benutzer, Login und Logout
components/	Bausteine, die auf mehreren Seiten verwendet werden, z.B. Dialoge
notifications/	Kurze Meldungen am unteren Bildschirmrand, z.B. «Satz gespeichert»
pages/	Ein File pro Seite, z.B. LoginPage.tsx

main.tsx startet die Anwendung. In App.tsx steht, welche Seite unter welchem Pfad angezeigt wird. Im Ordner public liegen die Icons und das Web-App-Manifest. Damit kann man die Anwendung auf dem Smartphone zum Home-Bildschirm hinzufügen.

7.3.7.2 Vite-Proxy

Lokal läuft das Frontend auf Port 5173 und das Backend auf Port 8080. Der Vite-Entwicklungsserver leitet alle Requests auf /api an das Backend weiter (siehe vite.config.ts).

Ohne diesen Proxy funktioniert der Login lokal nicht. Das Backend hat bewusst kein CORS, und das Cookie mit dem JWT ist HttpOnly und SameSite=Strict. Mit dem Proxy laufen Frontend und API lokal unter derselben Adresse, genau wie später hinter Caddy auf dem Server. Das Frontend kann das Token nicht lesen. Der Browser schickt das Cookie bei jedem Request selbst mit.

7.3.7.3 Material UI

Für das Aussehen wird die Komponentenbibliothek Material UI verwendet. Statt eigenem CSS kommen die Standardkomponenten wie Buttons, Dialoge und Listen zum Einsatz. Einstellungen für alle Komponenten stehen zentral in src/theme.ts, z.B. deutsche Texte und mindestens 44 Pixel hohe Buttons für die Bedienung mit dem Finger.

Die Schrift Roboto wird über @fontsource/roboto mitgeliefert und nicht von Google Fonts geladen. So lädt der Browser nichts von Drittanbietern.

7.3.7.4 Fehlermeldungen

Alle Requests an das Backend laufen über src/api/client.ts. Bei einem Fehler zeigt das Frontend einen deutschen Text passend zum HTTP-Statuscode an. Die englische Message vom Backend wird nie angezeigt.

Bei einem 401 ist die Sitzung abgelaufen und man wird automatisch abgemeldet. Ausnahme sind der Login und Aktionen mit Passwortbestätigung (z.B. Konto löschen). Dort kann 401 auch «Passwort falsch» heissen.

7.3.7.5 Dockerfile

Für das Frontend wurde ein Dockerfile erstellt. Es besteht aus zwei Stufen:

1. **Build:** Ein Node-Image installiert die Abhängigkeiten und erstellt mit npm run build die statischen Dateien.
2. **Runtime:** nginx liefert diese Dateien aus. Das fertige Image enthält weder Node noch node_modules.

Die Konfiguration von nginx steht in nginx.conf. Pfade wie /anmelden gibt es nur im Browser. Darum liefert nginx bei unbekannten Pfaden die index.html aus, und React Router zeigt die richtige Seite an. Lint und Tests laufen nicht im Dockerfile, sondern vorher in der GitHub-Pipeline.

7.3.8 Screenshots

Diese Screenshots zeigen auf, wie das fertige Produkt im Browser und auf dem Handy aussieht.

7.3.8.1 Desktop

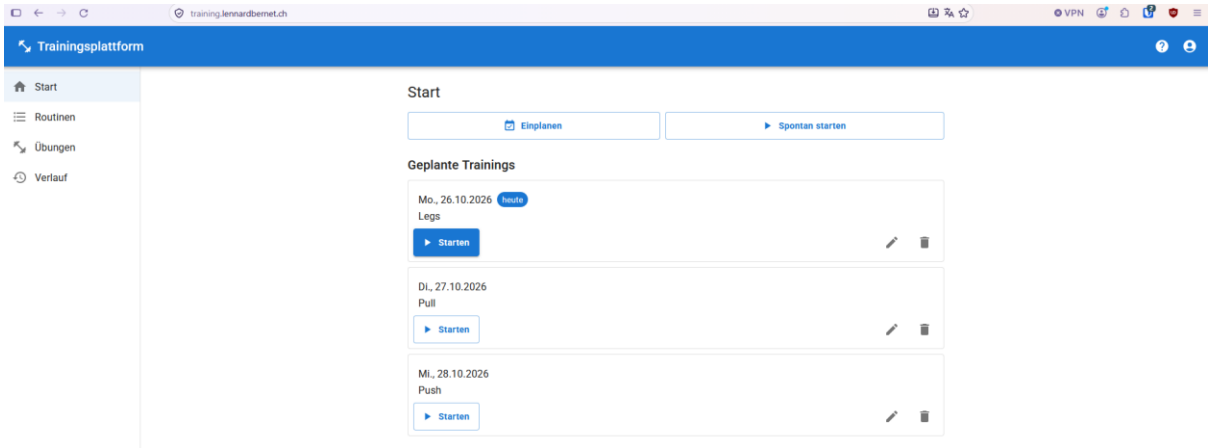


Abbildung 28: Startbildschirm

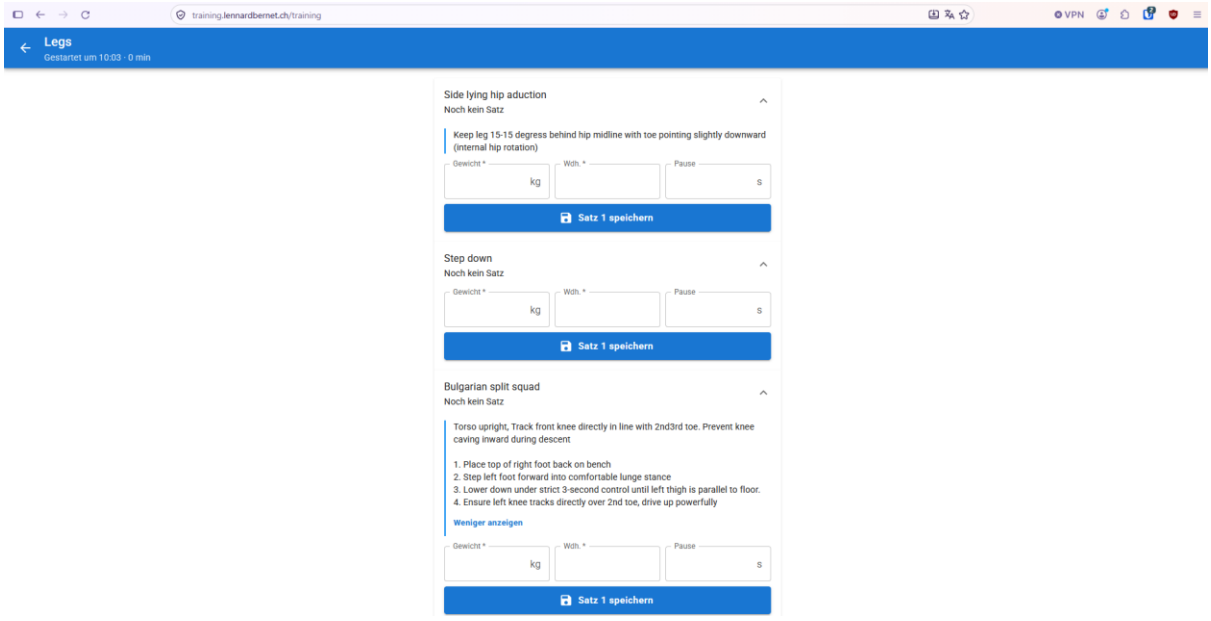


Abbildung 29: Live Training

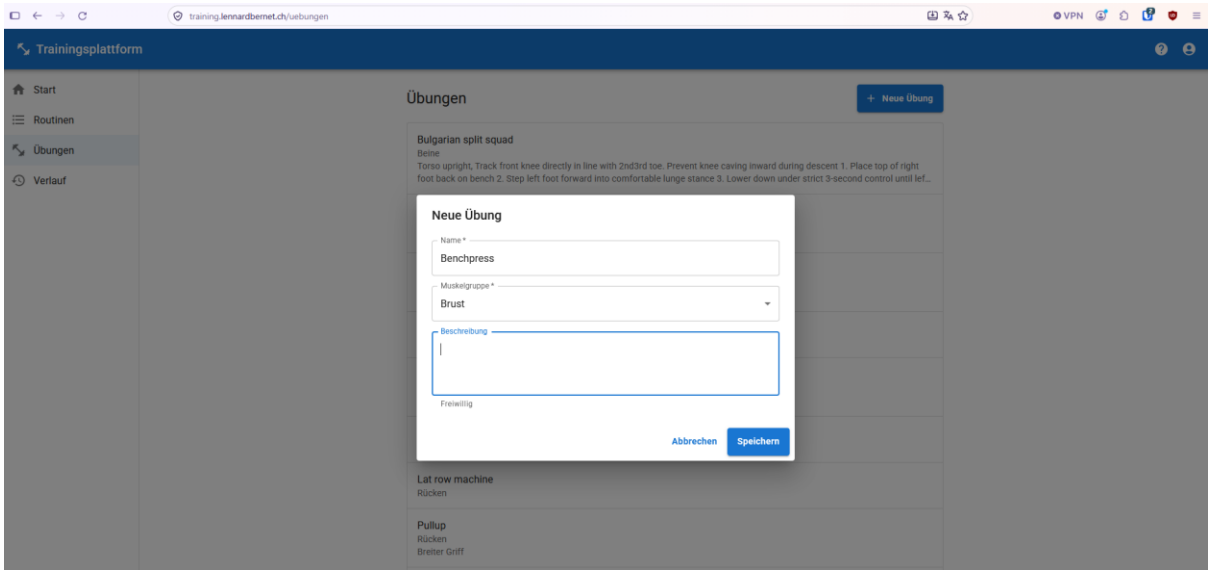


Abbildung 30: Neue Übung

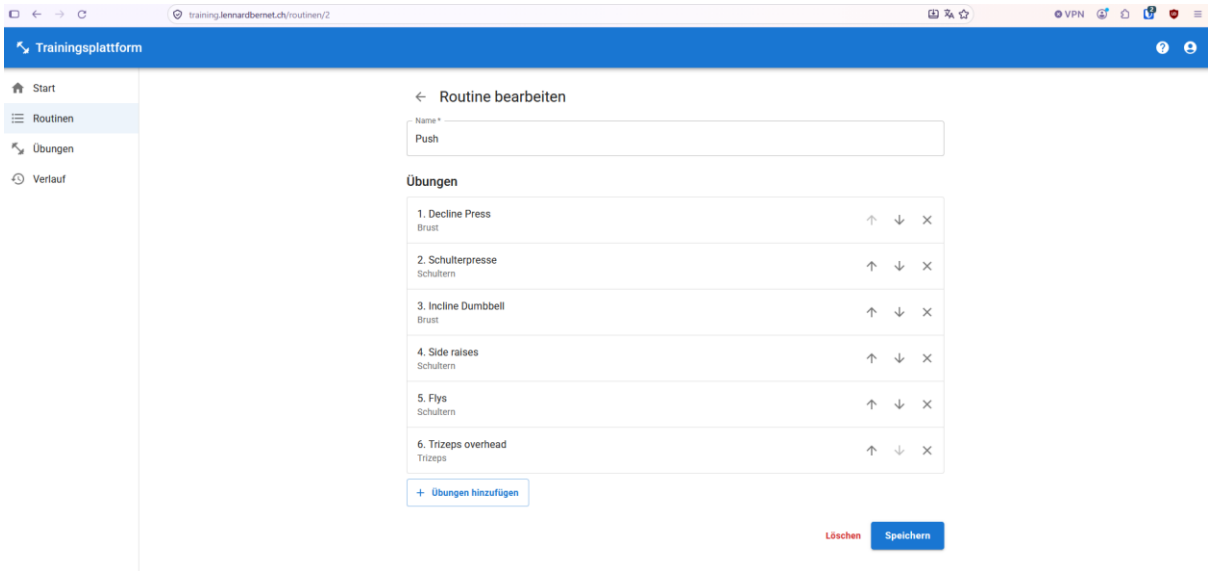


Abbildung 31: Routine bearbeiten

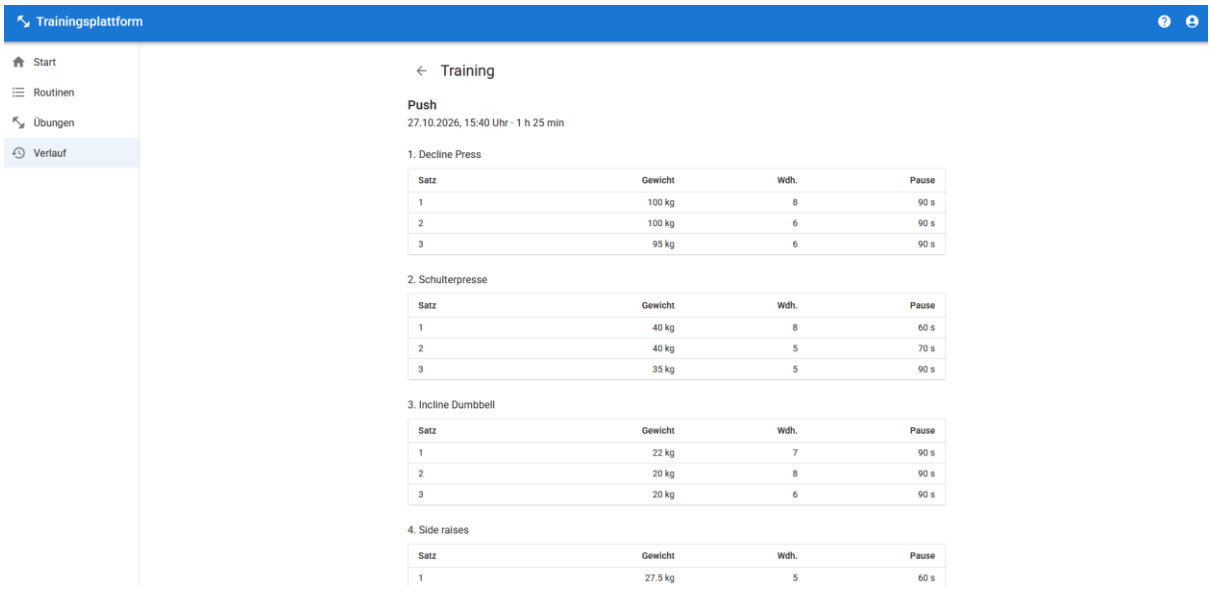


Abbildung 32: Verlauf

7.3.8.2 Mobile

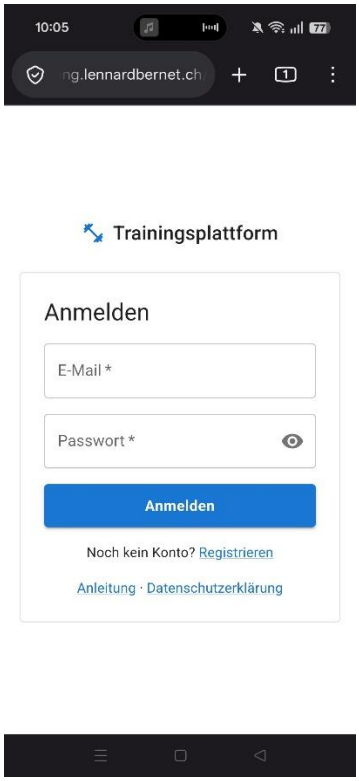


Abbildung 34: Anmelden

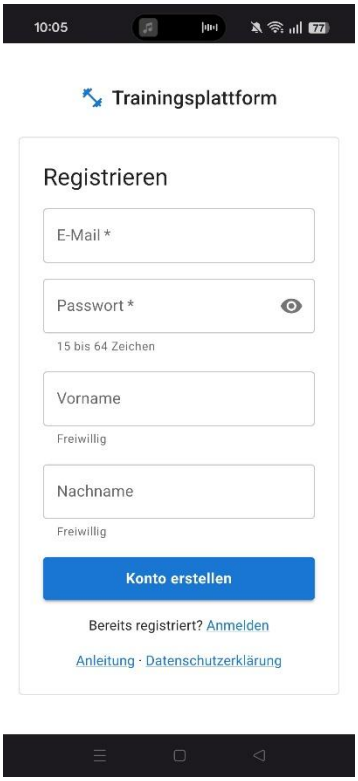


Abbildung 33: Registrieren

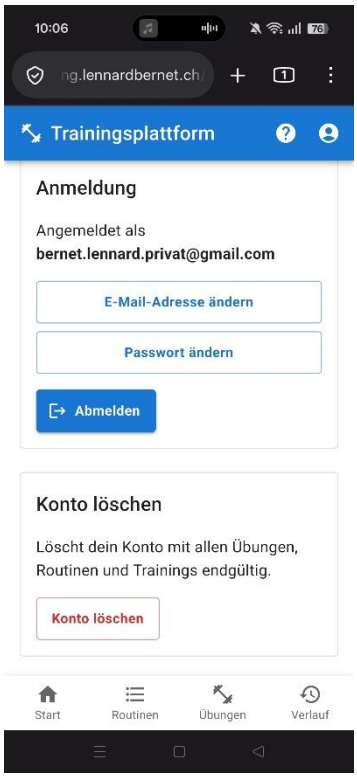


Abbildung 36: Konto

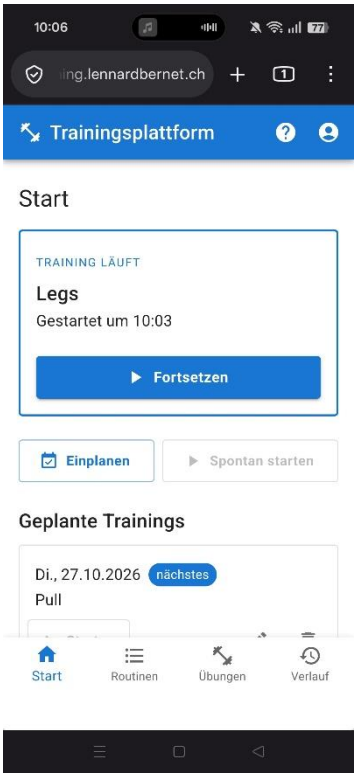


Abbildung 35: Übersicht

8 Deployment

Wie in der Deployment-Topologie und Technologie-Evaluation festgehalten wurde, wird ein einzelner Server bei Infomaniak aufgesetzt. Es handelt sich um einen VPS Lite mit 2 vCPU, 4 GB RAM und 60 GB Speicherplatz [22]. Das Rechenzentrum liegt in der Schweiz in Plan-les-Ouates. Als Betriebssystem wurde das Linux Ubuntu 26.04 LTS 64-Bit gewählt. Beim Aufsetzen wurde ein neues SSH-Zertifikat generiert, mit dem man sich Lokal oder von GitHub aus mit dem Server verbinden kann.

Direkt auf der Website von Infomaniak gibt es einen Firewall Manager, bei dem man die offenen Ports konfigurieren kann. Port 22 ist für SSH schon offen. Neu geöffnet wurden noch 80 und 443 TCP für die Verbindung per HTTP/HTTPS. Der Port 5432 für die Postgres-Datenbank darf nicht geöffnet werden. Die Datenbank soll nur intern im Docker-Netz erreichbar sein.

Die IP-Adresse des Servers wird auf die eigene Domain gesetzt: training.lennardbernet.ch.

8.1 Reverse Proxy

Ein Reverse Proxy ist der Türsteher für die Dienste, die auf dem Server laufen. Der Browser spricht nur mit dem Reverse Proxy, nie direkt mit dem Backend oder Frontend. Als Reverse Proxy in dieser Diplomarbeit wird Caddy verwendet [23]. Er schaut sich jede Anfrage an und leitet sie an den richtigen Dienst weiter. Die Antwort an den Browser geht denselben Weg zurück. Dies wurde auch schon in der Bausteinsicht bei der Definition der Systemarchitektur so festgehalten.

Der Reverse Proxy erlaubt es, dass das Frontend und Backend über dieselbe Domäne laufen. Somit hat man keine Probleme mit CORS und das SameSite=Strict-Cookie funktioniert.

Der Caddy holt sich das HTTPS-Zertifikat selbst bei Let’s Encrypt und erneuert es selbst. Anfragen, die schon mit HTTPS reinkommen, werden entschlüsselt und laufen unverschlüsselt ins Docker-Netz. Die Antworten werden dann aber immer verschlüsselt zum Browser zurückgeschickt.

Das Backend, Frontend und die DB sind von aussen nicht erreichbar. Die Container hinter dem Reverse Proxy können neu starten oder sich ändern und die Adresse bleibt gleich.

Der Caddy wird über das Caddyfile konfiguriert. Dies muss von Hand auf den Server kopiert werden.

8.2 Docker Compose

Docker Compose ist ein Bauplan für mehrere Container. Die Konfiguration beschreibt, welche es gibt, wie sie verbunden sind und welche Einstellungen sie jeweils bekommen. In diesem Fall werden 4 Services gebraucht: Der Caddy Reverse Proxy, das eigene Backend und Frontend sowie eine Postgres-Datenbank.

8.2.1 Reverse Proxy

Wie im vorherigen Kapitel beschrieben, wird der Caddy als Reverse Proxy gebraucht. Das Image dafür wird von einem bestehenden Dockerhub Repository gezogen.

Die Ports 80 und 443 werden geöffnet. Kein einziger anderer Service im Docker-Compose-File hat offene Ports.

Die Domain wird über das lokale .env File auf dem Server eingelesen. In diesem Fall ist es training.lennardbernet.ch.

Mit den festen Volumes caddy_data und caddy_config überleben die Zertifikate einen Neustart. Ohne sie würde Caddy bei jedem Neustart ein neues Zertifikat holen, was die Rate-Limits von Let's Encrypt schnell aufbrauchen würde.

Der Caddy ist im Netz «Web» und darf nur mit dem Frontend und Backend reden.

8.2.2 Frontend

Das Frontend Image wird vom eigenen Dockerhub Repository gezogen. Der Tag wird über die Umgebungsvariable ${IMAGE_TAG} gesetzt. Ist diese nicht vorhanden, wird das latest Image gezogen.

Das Frontend hat keine Ports. Es ist im Netz «Web» und darf somit nur mit dem Caddy und Backend reden.

8.2.3 Backend

Wie das Frontend wird auch das Backend vom eigenen Dockerhub Repository gezogen. Der Tag wird über dieselbe Umgebungsvariable gesetzt. Zusätzlich kommen noch das DB_PASSWORD sowie das JWT_SECRET aus der .env Datei auf dem Server.

Das Backend ist abhängig von der Datenbank, denn Flyway braucht sie beim Start. Dafür wird geprüft, ob die Datenbank den Status «service_healthy» hat.

Es hängt als einziger Service in zwei Netzen: «Web» und «Db». Somit ist es der einzige Service, der mit der Datenbank reden darf.

8.2.4 Datenbank

Als Datenbank wird ein Postgres-Image vom Dockerhub gezogen.

Es hängt im Netz «Db» und kann somit nur vom Backend angesprochen werden.

Ein Healthcheck prüft alle 5 Sekunden mit «pg_isready», ob die Datenbank bereit ist

Mit dem Volume «db_data» wird wie beim Reverse Proxy sichergestellt, dass die Daten bei einem Neustart nicht verloren gehen.

8.2.5 Logging

Alle Container schreiben die Logs an einen zentralen Ort auf dem Server. Es ist so eingestellt, dass die Einträge täglich in eine neue Datei kommen und nach 14 Tagen gelöscht werden.

8.3 Deploy Workflow

Für die Automatisierung wird eine kleine CI/CD Pipeline erstellt. Das ist eine normale Grundlage im modernen Softwareentwicklungsprozess.

Im GitHub Repository wurde dafür eine Workflow erstellt, welcher immer ausgeführt wird, sobald man einen Pull Request auf den Main Branch gemerged hat. Der erste Job ist für das Erstellen der Dockerhub Images. In den GitHub Secrets ist der Dockerhub Username sowie ein API-Token hinterlegt. Somit können die gebauten Images direkt ins richtige Repository gepusht

werden. Als Image Tag wird der SHA-Tag des Commits, der durch den Merge auf main entstanden ist, verwendet. Somit hat man eindeutig erkennbare Images.

Der zweite Job im Workflow ist für den Deploy auf dem VPS Lite Server. In den GitHub Secrets ist die IP-Adresse sowie der Private Key des SSH-Schlüssels hinterlegt. Somit kann eine sichere Verbindung aufgebaut werden. Der gleiche SHA-Tag des Commits wird nun als Umgebungsvariable gesetzt. Weil das docker-compose File die Tags aus den Umgebungsvariablen liest, werden immer die richtigen Images angezogen. Die Plattform ist für einen kurzen Moment nicht erreichbar, da kein Cluster vorhanden ist mit mehreren Pods. Das ist aber vollkommen in Ordnung so und kein Problem.

Nach dem Deploy prüft der Workflow über den Health-Check vom Backend ob es wieder läuft. Ist dies innerhalb von 5 Minuten nicht der Fall, wird der Workflow rot.

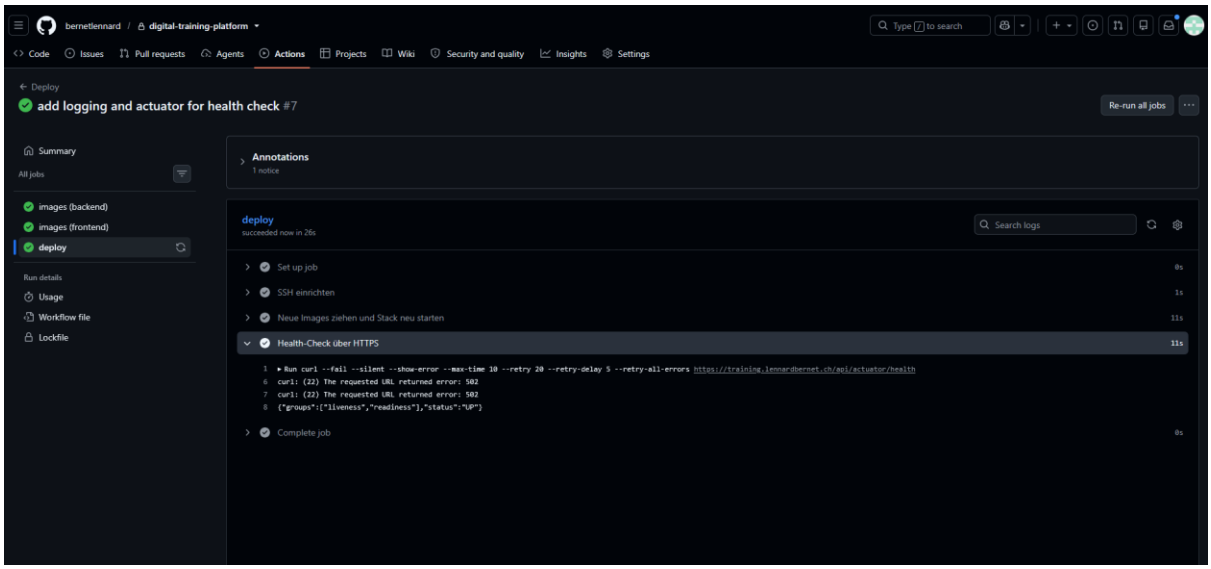


Abbildung 37: Deploy Workflow

8.4 Smoke-Tests

Die Smoke-Tests müssen nach jedem Deployment gegen training.lennardbernet.ch ausgeführt werden. Sie prüfen, ob die Services korrekt konfiguriert und erreichbar sind oder eben nicht.

Tabelle 97: Smoke-Tests

ID	Testfall	Erwartetes Ergebnis
SMK-01	GitHub → «Actions» → Workflow «Deploy» des Merges	Jobs «images» und «deploy» grün
SMK-02	Auf dem Server docker compose ps	caddy, frontend, backend und db laufen, db «healthy»
SMK-03	Auf dem Server docker compose images	frontend und backend mit dem Tag des Merge-Commits
SMK-04	http://training.lennardbernet.ch im Browser	Umleitung auf HTTPS, Zertifikat gültig
SMK-05	https://training.lennardbernet.ch	Anmeldeseite erscheint
SMK-06	/verlauf direkt aufrufen	Anmeldeseite statt Fehlerseite
SMK-07	/api/v1/users/me ohne Anmeldung	401 im JSON-Fehlerformat
SMK-08	/swagger-ui.html	Keine Swagger-Oberfläche

		Anmeldeseite erscheint
SMK-09	In der PowerShell auf dem eigenen Rechner: Test-NetConnection training.lennardbernet.ch -Port 5432 Dasselbe mit -Port 8080	Beide Male TcpTestSucceeded : False Datenbank und Backend von aussen nicht erreichbar
SMK-10	Mit dem Smoke-Test-Konto anmelden	Startseite erscheint
SMK-11	«Verlauf» öffnen	Training vom letzten Mal noch vorhanden
SMK-12	Übung anlegen und wieder löschen	Beides mit Bestätigung, Übung danach weg
SMK-13	«Konto» → «Abmelden»	Anmeldeseite

9 Benutzeranleitung

9.1 Überblick

Mit der Trainingsplattform verwaltest du deine Übungen und Routinen, planst deine Trainings und protokollierst sie direkt im Training. Zuerst erfasst du deine Übungen und stellst daraus eine Routine zusammen. Diese Routine kannst du danach einplanen oder spontan starten. Im Training erfasst du deine Sätze und nach dem Abschliessen findest du alles im Verlauf.

Die Plattform ist für das Smartphone gemacht. Am unteren Rand ist die Navigation mit Start, Routinen, Übungen und Verlauf. Auf einem Tablet oder Computer ist sie als Seitenleiste links. Oben rechts kommst du zu dieser Anleitung und zu deinem Konto.

9.2 Registrieren und anmelden

Für die Plattform brauchst du ein Konto. Unter «Registrieren» gibst du deine E-Mail-Adresse und ein Passwort ein. Vor- und Nachname sind freiwillig. Nach «Konto erstellen» bist du direkt angemeldet.

Danach meldest du dich unter «Anmelden» mit E-Mail-Adresse und Passwort an. Die Anmeldung gilt für acht Stunden, danach musst du dich neu anmelden. Nach mehreren falschen Passwörtern wird die Anmeldung zum Schutz für 15 Minuten gesperrt.

9.3 Übungen erfassen

Unter «Übungen» baust du deinen eigenen Übungskatalog auf. Mit «Neue Übung» gibst du den Namen ein und wählst die Muskelgruppe. In der Beschreibung kannst du Hinweise zur Ausführung notieren, zum Beispiel die Einstellung am Gerät. Diese Beschreibung siehst du später auch im Training.

Tippst du eine Übung an, kannst du sie ändern oder löschen. Kommt die Übung noch in einer Routine vor, musst du sie zuerst dort entfernen. Abgeschlossene Trainings mit dieser Übung bleiben im Verlauf vollständig erhalten.

9.4 Routinen zusammenstellen

Eine Routine ist dein Trainingsprogramm mit mehreren Übungen, zum Beispiel für den Oberkörper. Unter «Routinen» erstellst du mit «Neue Routine» eine neue Routine. Gib einen Namen ein und wähle über «Übungen hinzufügen» die Übungen aus. Sie kommen in der Reihenfolge in die Routine, in der du sie antippst. Mit den Pfeilen änderst du die Reihenfolge und mit dem Kreuz entfernst du eine Übung wieder. Nach «Speichern» ist die Routine bereit.

Löschst du eine Routine, werden auch ihre geplanten Trainings gelöscht. Abgeschlossene Trainings bleiben im Verlauf.

9.5 Training planen

Auf der Startseite planst du mit «Einplanen» ein Training. Dafür wählst du die Routine und das Datum, frühestens heute. Die geplanten Trainings stehen nach Datum sortiert auf der Startseite und das nächste ist markiert. Mit dem Stift änderst du ein geplantes Training und mit dem Papierkorb löschst du es.

Ein verpasstes Training wird als «vergangen» angezeigt. Du kannst es noch starten oder löschen, aber nicht mehr verschieben.

9.6 Training durchführen

Ein geplantes Training startest du auf der Startseite mit «Starten». Ohne Planung tippst du auf «Spontan starten» und wählst die Routine aus. Es kann immer nur ein Training gleichzeitig laufen.

Oben siehst du die Startzeit und wie lange das Training schon läuft. Jede Übung hat einen eigenen Bereich, den du mit einem Tipp auf- und zuklappst. Für einen Satz gibst du das Gewicht in kg, die Wiederholungen und freiwillig die Pause in Sekunden ein. Mit «Satz 1 speichern» ist der erste Satz erfasst.

Hast du die Routine schon einmal trainiert, sind die Felder mit den Werten vom letzten Training vorbelegt. Die weiteren Sätze stehen als «offen» darunter. Somit musst du die Werte nur noch anpassen oder direkt speichern. Sind keine Sätze mehr offen, erscheint bei der Übung ein grünes Häkchen.

Tippst du einen gespeicherten Satz an, kannst du die Werte ändern, den Satz an eine andere Position verschieben oder ihn löschen. Unten gibt es ein Feld für eine Notiz zum Training. Sie wird gespeichert, sobald du das Feld verlässt.

Mit dem Pfeil oben links kommst du zur Startseite, das Training läuft dabei weiter. Mit «Fortsetzen» bist du wieder im Training. Auch wenn du die Seite schliesst, geht nichts verloren.

Am Schluss tippst du auf «Abschliessen». Dafür braucht es mindestens einen Satz. Übungen ohne Satz werden nicht gespeichert und das Training lässt sich danach nicht mehr ändern. Mit «Abbrechen» werden alle Sätze verworfen. Ein geplantes Training bleibt dabei eingeplant.

9.7 Verlauf ansehen

Unter «Verlauf» findest du deine abgeschlossenen Trainings nach Routine geordnet. Tippst du eine Routine an, siehst du ihre Trainings mit Datum und Dauer, das neuste zuoberst. Im Training selbst stehen alle Übungen mit ihren Sätzen und der Notiz. So siehst du, wie du dich von Mal zu Mal steigerst.

Mit «Training löschen» entfernst du ein Training aus dem Verlauf. Gelöschte Routinen bleiben im Verlauf und sind als «gelöscht» markiert.

9.8 Konto verwalten

Über das Symbol mit der Person oben rechts kommst du zu deinem Konto. Dort änderst du deinen Namen, deine E-Mail-Adresse und dein Passwort. Bei der E-Mail-Adresse und beim Passwort musst du zur Sicherheit dein aktuelles Passwort eingeben. Mit «Abmelden» meldest du dich ab, was vor allem auf einem fremden Gerät wichtig ist.

Mit «Konto löschen» werden dein Konto und alle Trainingsdaten nach der Bestätigung mit dem Passwort endgültig gelöscht. Welche Daten die Plattform speichert, steht in der Datenschutzerklärung.

10Abschluss

10.1 Testergebnisse

Vor der Abgabe wurde der finale Stand per Pull Request auf main gemergt, wodurch alle automatisierten Tests in der Pipeline gelaufen sind. Die manuellen Tests wurden lokal oder gegen die Produktion ausgeführt.

Tabelle 98: Testergebnisse

Testgruppe	Anzahl	Durchführung	Datum	Resultat
Unit-Tests Backend	204	GitHub Actions	27.10.2026	Alle erfolgreich
Unit-Tests Frontend	71	GitHub Actions	27.10.2026	Alle erfolgreich
Systemtests Backend	3	Manuell, IntelliJ HTTP-Client	27.10.2026	Alle erfolgreich
Systemtests Frontend	52	Manuell, lokal	27.10.2026	Alle erfolgreich
Status Trainingseinheit	14	Manuell, lokal	27.10.2026	Alle erfolgreich
Löschkonzept	17	Manuell lokal	27.10.2026	Alle erfolgreich
Smoke-Tests	13	Manuell, Produktion	27.10.2026	Alle erfolgreich

10.2 Erfolgskontrolle

Als erstes möchte ich zurückblicken auf den Projektbeschrieb. Genauer gesagt auf die vorgegebenen Muss-Ziele. Diese sollten die Grundlage dieser Software bilden. Mit Stolz kann ich sagen, dass alle Muss-Ziele erfolgreich erfüllt wurden. Dadurch, dass schon ziemlich früh festgelegt wurde, dass ich mich auf diese Ziele begrenze, hatte ich bei der Anforderungsanalyse ein gutes Gefühl dabei. Wenn ich jetzt auf die Kann-Ziele schaue, ist es klar, dass noch einiges zu tun ist, bis diese sauber umgesetzt sind. Durch die klare Abgrenzung wurde auch noch nichts für diese Ziele gemacht.

Mit dem Blick auf die Projektvision würde ich meinen, dass das finale Produkt genau diese Elemente abdeckt. Der komplette Zyklus eines Krafttrainings kann festgehalten werden. Man hat keinen Überfluss von Features, sondern es ist genau auf das begrenzt, was ich persönlich für eine saubere Protokollierung im Fitnesscenter brauche. Ich weiss, dass alles technisch verlässlich ist und bin wirklich zufrieden mit der Bedienbarkeit. Das Ziel von der Freiheit einer Open-Source-Lösung mit dem Bedienkomfort einer kommerziellen Software wurde sicherlich erfüllt.

Ebenfalls wurden die Erfolgskriterien aus der Themeneingabe alle erfüllt. Hier sind diese nochmals aufgelistet mit einer Erfolgskontrolle:

Tabelle 99: Erfolgskontrolle

Erfolgskriterien	Erfolgskontrolle
Ein authentifizierter Benutzer kann eine individuelle Trainingsroutine aus mindestens drei Übungen erstellen, bearbeiten und für ein zukünftiges Datum in einen Kalender einplanen.	Die Authentifizierung wird mit dem JWT sichergestellt. Trainingsroutinen sind auf individuelle Benutzer eingeschränkt. Es können theoretisch unendlich viele Übungen erstellt werden. Die Planung der Trainingsroutinen ist möglich.
Während einer aktiven Trainingseinheit können Gewicht, Wiederholungen und Pausenzeiten pro Satz direkt protokolliert und gespeichert werden.	Genau diese Informationen können live protokolliert werden und werden auf der Datenbank persistiert.
Die Benutzeroberfläche ist vollständig responsiv. Das Frontend ist auf dem Smartphone sowie auf einem Desktop möglich zu bedienen.	Das Layout im Frontend wurde für Smartphones entwickelt und passt sich an bei grösseren Bildschirmen.
Die Geschäftslogik des Backend ist durch automatisierte Unit, und Integrationstests abgesichert.	Anhand der definierten Anforderungen wurden die Testfälle geschrieben, welche automatisiert laufen oder klare Anweisungen für die manuelle Prüfung haben.
Das Backend bietet eine klar definierte und dokumentierte Schnittstelle für das Frontend. Für sämtliche REST-Endpunkte wird eine interaktive API-Dokumentation mit Swagger generiert die im Browser aufrufbar ist.	Im Konzept ist die REST-Schnittstelle mit allen Request und Response DTOs festgehalten. Die Swagger Dokumentation ist vollständig, aber nur lokal erreichbar.
Eine tokenbasierte Authentifizierung und Autorisierung mit Spring Security sichert den Datenzugriff im Backend. Diese Mechanismen sind ebenfalls mit automatisierten Tests abgesichert.	Die Autorisierung und Authentifizierung mit dem Spring Security JWT ist ausführlich automatisiert getestet.
Ein authentifizierter Benutzer kann sein komplettes Profil und sämtliche Daten selbstständig über die Benutzeroberfläche löschen. Die Daten werden dabei restlos aus der Datenbank entfernt.	Die Löschregeln auf der Datenbank stellen sicher, dass alle Daten von einem Benutzer hart gelöscht werden.
Der Deployment-Prozess der Applikation ist vollständig automatisiert. Das Pushen von neuem Code im Git stösst eine CI/CD-Pipeline an. Diese verpackt die Applikation als Docker-Container und rollt die neue Version vollautomatisiert auf der PaaS Infrastruktur aus.	Mit den erstellten GitHub Actions wird genau dieses Kriterium erfüllt. Einzige Abweichung ist, dass hier die Lösung von dem VPS bei Infomaniak streng genommen kein PaaS ist. Mit den Entscheidungen E4 und E7 ist aber gut belegt, warum die gewählte Lösung für die Diplomarbeit die passende ist.

10.3 Projektrisikomanagement

So ein ausführliches Projektrisikomanagement war eine ziemliche Herausforderung direkt am Anfang zu machen. Natürlich habe ich solche Risikomatrizen auch schon gesehen, diese aber selbst zu entwickeln war eine lehrreiche Erfahrung.

Die getroffenen Massnahmen haben mich über das ganze Projekt hinweg sauber in Schach gehalten. Sie gaben einen strukturierten Fahrplan vor, welcher ein Gefühl der Sicherheit gab, da die Risiken bekannt waren. Die Massnahmen für das Frontend und Deployment habe ich zu Herzen genommen. Dort habe ich am meisten Zeit gespart, um die Diplomarbeit sauber abzuschliessen. Aber auch das Risiko der persönlichen Verfügbarkeit hat sich während dem Projekt gezeigt. Im September gab es einen Zeitraum, wo genau solche unerwarteten privaten und beruflichen Belastungen stattgefunden haben. Dadurch geriet ich in einen Rückstand, welcher den Zeitplan im Oktober extrem gedrückt hat. Spätestens dann war klar, dass ich mich voll auf die Muss-Ziele fokussieren muss, um wenigstens diese sauber über die Ziellinie bringen.

10.4 Feedback Auftraggeber

Laut Themeneingabe gibt es keinen Auftraggeber, da ich selbst der Nutzer dieser Plattform bin. Bis zum Zeitpunkt der Abgabe habe ich schon einige Trainings mit der Unterstützung der Plattform durchgeführt und bin wirklich zufrieden mit dem Endresultat.

10.5 Reflexion

Rückblickend war die Erarbeitung dieser Diplomarbeit ein grosser Lernprozess für mich persönlich. Während dem ganzen Projekt konnte ich mein Wissen der Softwareentwicklung in vielen verschiedenen Aspekten anwenden und erweitern. Ich habe dabei die Rolle als Projektmanager, Softwarearchitekt, Requirement-Engineer und klassischer Entwickler eingenommen. Wie schon angesprochen war das Projektrisikomanagement wirklich wichtig und hat sich bewährt. Auch die Wirtschaftlichkeitsanalyse hat Spass gemacht, um mal genauer zu schauen, was die Kosten für so ein Softwareprojekt einer echten Firma wären.

Die Planung an sich war gut bedacht. Man sieht aber direkt im Gantt-Diagramm, dass der Zeitplan ab dem September nicht eingehalten wurde. Grund dafür waren die Beschwerden und Schmerzen im Handgelenk, welche dazu führten, dass ich schlichtweg nicht mehr richtig arbeiten konnte. Im Oktober gingen die Symptome wieder zurück und ich musste Vollgas geben. Schlussendlich bin ich jetzt auch einfach erleichtert, dass alles funktioniert hat und es keine Fristerstreckung braucht.

Rückblickend wäre ein anderer Projektmanagement-Ansatz auch möglich gewesen. Spätestens bei der Entwicklung hätten Sprints über die jeweiligen Fachbereiche Benutzerverwaltung, Routinen, Übungen und Trainingsverlauf Sinn gemacht. Hier habe ich mich aber einfach an die vordefinierte Planung gehalten, da es ansonsten ein riesiges Chaos gegeben hätte.

Die tiefgründige Analyse der verschiedensten Technologien und Architekturansätze hat wirklich Spass gemacht. Ich hatte im Kopf zwar schon eine Vorstellung, was ich machen will und es war gar nicht so einfach, diese Entscheidungen ohne Persönliche Meinung zu treffen. Beim Cyber Security Konzept war auch das Abarbeiten der OWASP Top 10 Liste eine interessante Erfahrung. Gerne würde ich mehr Erfahrung in Cyber Security Themen haben.

Bei der Anforderungsanalyse konnte ich dann endlich wieder die Themen aus der Projektvision so richtig angreifen. Da die Plattform ja schlussendlich ziemlich sicher nur von mir gebraucht wird, wollte ich auch sicherstellen, dass genau meine Bedürfnisse abgedeckt werden. Darum ist dieser Teil der Diplomarbeit mir auch ziemlich leichtgefallen. Das ganze aber in Anforderungen zu verpacken, welche alle miteinander harmonisieren, war gar nicht so einfach. Es ist aber schlussendlich durch den strukturierten Aufbau von Geschäftsanwendungsfällen, Systemanwendungsfällen und Features gut gelungen.

Noch mehr Freude hatte ich dann, wenn es endlich Richtung Konzeption und Entwicklung ging. In diesen Bereichen bringe ich am meisten Erfahrung und sie machen mir auch einfach Spass. Den Ansatz, mit dem ERM zu beginnen und dann über das Backend zum Frontend zu gehen, habe ich schon in verschiedensten Softwareprojekten so gemacht. Dies hat sich auch hier wirklich bewährt. Während der Entwicklung konnte ich dann auf das sauber ausgearbeitete Konzept schauen und zusammen mit den Anforderungen und Technologieentscheidungen war schnell klar, dass ich komplett auf dem richtigen Weg war.

Die Entscheidung, im Frontend stark auf Claude Code zu setzen, bereue ich leider schon etwas. Wäre ich nicht so eng im Zeitplan gewesen, hätte ich hier mehr Zeit investiert. Frontendentwicklung ist definitiv nicht meine Stärke und diese Diplomarbeit wäre eine gute Gelegenheit gewesen, um hier mein Skillset zu erweitern. Natürlich wird es immer gängiger in diesem Beruf, dass man KI als Hilfsmittel einsetzt, aber die Grundlagen muss man trotzdem beherrschen.

Überraschend gut hat das ganze Deployment funktioniert. Durch das Fach «Containerisierung», welches wir im letzten Semester hatten, kannte ich den ganzen Ablauf schon und konnte das Ganze eigentlich einfach kopieren und leicht anpassen. Hier habe ich mit mehr Hürden gerechnet.

Ich werde persönlich sicher weiter an diesem Projekt arbeiten. Es wurde schon so viel Zeit darin investiert und es besteht auch ein Produkt, welches ich im Fitnessstudio selbst brauche. Die Kann-Ziele werden ziemlich sicher bald angegangen. Durch die saubere Aufteilung im Backend werden die geplanten Erweiterungen absolut kein grosses Problem darstellen. Ich freue mich schon darauf.

10.6 Lessons learnt

Der Entwicklungsansatz mit TDD hat sich richtig gelohnt. Das hat sich gezeigt, als die Satzvorschläge umgebaut wurden. Die neu geschriebenen Tests zeigten sofort, was nicht mehr stimmt. Der Aufwand, um die Tests am Anfang zu machen, lohnt sich wirklich und sollte auch bei kommenden Projekten so gemacht werden.

Was jedoch etwas fehlt, sind automatisierte Integrationstests mit echter Datenbank. Die aktuelle Lösung mit den HTTP-Requests aus dem IntelliJ oder die manuelle Systemtests aus dem Frontend laufen halt nicht bei den Pipelines. Beim nächsten Projekt oder im Anschluss sollte man Testcontainer dafür verwenden.

Dass sich die erste Version der Satzvorschläge falsch anfühlt, hat man erst beim Bedienen gemerkt. Die Lehre daraus ist, einen bedienbaren Stand schon früh aufs Handy zu bringen und im Training auszuprobieren. Hier spielt auch das erkannte Risiko Bias beim Testing mit. Nur weil es für mich jetzt so passt, ist es für andere nicht automatisch intuitiv und nachvollziehbar, von wo die Satzdaten kommen.

In der Planung sollten die Puffer noch konkreter eingeplant werden. Am Anfang wurde genau dieses Risiko beschrieben und man hat schon gewisse Pufferstunden eingeplant pro Arbeitspaket. Noch besser wäre es, jeweils Pufferwochen zwischen den Projektphasen einzuplanen. Somit hätte ich im Juli/August schon mehr geleistet und wäre nicht so krass im Verzug gewesen.

10.7 Dankeschreiben

Ich möchte mir hier einen kurzen Moment nehmen, um mich bei allen Beteiligten während meiner Zeit an der TEKO Zürich zu bedanken. Diese vorliegende Diplomarbeit gelang nur durch die Zusammenarbeit von vielen Personen: den Dozierenden, die mich, während der letzten 6 Semester weitergebildet haben, und dem Sekretariat, welches die organisatorischen Aufgaben immer im Hintergrund gemeistert hat.

Meinem Diplomcoach Dominic Merz danke ich für seinen grossartigen Unterricht sowie hilfreiche Unterstützung während dieser Diplomarbeit. Ich konnte vieles Gelernte aus seinem Unterricht und vorherigen Transferarbeiten hier in der Diplomarbeit wiederverwenden.

Ebenfalls bedanke ich mich bei all meinen Mitschülern, mit denen ich viel gelacht und gelernt habe. Es war eine tolle Zeit mit guten Erinnerungen.

10.8 Kommentar zu KI

Wie die Richtlinie zur Nutzung von KI an der TEKO Zürich vorgeschrieben hat, wurden die nötigen Stellen klar gekennzeichnet direkt im Text mit der dafür vorgesehenen braunen Farbe. Dies ist der Fall für den Buy-Case [5], der Absatz zu den Gesundheitsdaten [14], die Wissensabfrage über PostgreSQL [16] und das Glossar.

An folgenden weiteren Stellen in der Diplomarbeit wurde das Ergebnis mit Claude Code Opus 5.5 erarbeitet: Frontend-Code, die Vitest-Tests und die HTTP-Prüfungen. Dies ist jeweils ebenfalls im Text klar notiert.

Ansonsten wurde KI im Rahmen dieser Diplomarbeit ausschliesslich zur stilistischen und grammatikalischen Überarbeitung einzelner Textpassagen eingesetzt. Sie diente der Verbesserung der Lesbarkeit, der Vereinheitlichung des Schreibstils und der Kontrolle der sprachlichen Einheitlichkeit zwischen den Kapiteln.

11 Anhang

11.1 Eigenständigkeitserklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

Alle sinngemäss und wörtlich übernommenen Textstellen aus fremden Quellen wurden kenntlich gemacht.

8180 Bülach, den 28. Oktober 2026 Unterschrift: 

11.2 Glossar

Für das Glossar wurde der gesamte Kontext dieser Arbeit an Claude Opus 5.5 übergeben. Mit folgendem Prompt wurde der Inhalt dieses Kapitels erstellt:

«Erstelle aus dieser Diplomarbeit ein Glossar mit den Fachbegriffen und Abkürzungen, die im Text vorkommen. Erkläre jeden Begriff kurz und verständlich, wo es passt mit Bezug auf diese Arbeit. Gib das Glossar als alphabetisch sortierte Tabelle mit den Spalten Begriff und Beschreibung aus.»

Anschliessend wurde die Liste gründlich geprüft, um sicherzustellen, dass die Einträge im Glossar meiner Vorstellung entsprechen. Da alles gepasst hat, wurde es so übernommen und hier eingefügt:

Tabelle 100: Glossar

Begriff	Beschreibung
Akteur	Person oder System, das mit der Plattform arbeitet, hier Besucher, Sportler und Betreiber.
API	Application Programming Interface. Schnittstelle, über die zwei Programme miteinander kommunizieren.
API-Versionierung	Die Version steht im Pfad, z.B. /api/v1. Inkompatible Änderungen kommen in eine neue Version.
Architekturtreiber	Ziel oder Rahmenbedingung, welche die Architektur stark beeinflusst. Kriterien der Entscheidungsmatrizen.
Argon2id	Von OWASP empfohlenes Verfahren zum Hashen von Passwörtern.
Authentifizierung	Prüfung, wer ein Benutzer ist, hier beim Login und danach bei jedem Request über das JWT.
Autorisierung	Prüfung, was ein angemeldeter Benutzer darf. Ein Sportler sieht nur seine eigenen Daten.
Backend	Teil der Applikation auf dem Server mit der Geschäftslogik, der REST-Schnittstelle und dem Datenbankzugriff.
Bausteinsicht	Darstellung der Architektur mit den grossen Bausteinen des Systems, ohne einzelne Klassen.
Besitzerprüfung	Das Backend prüft bei jedem Zugriff anhand der Benutzer-ID aus dem Token, ob die Daten dem Sportler gehören.

Brute-Force-Attacke	Automatisiertes Ausprobieren vieler Passwörter. Dagegen schützt die Login-Sperre.
Buy-Case und Make-Case	Vergleich, ob man eine bestehende App kauft (Buy) oder die Software selbst entwickelt (Make).
Caddy	Webserver, der hier als Reverse Proxy dient und die HTTPS-Zertifikate automatisch holt und erneuert.
CASCADE	Löschregel bei einem Fremdschlüssel. Abhängige Datensätze werden automatisch mitgelöscht.
CI/CD	Continuous Integration und Continuous Deployment. Code wird automatisch gebaut, getestet und ausgeliefert.
Constraint	Regel in der Datenbank, welche die Daten absichert, z.B. NOT NULL, UNIQUE, CHECK oder ein Fremdschlüssel.
Container	Abgeschottete Umgebung, in der eine Anwendung mit allem läuft, was sie braucht.
Container-Image	Vorlage für einen Container. Die Version wird mit einem Tag gekennzeichnet, hier dem SHA des Merge-Commits.
Container-Orchestrierung	Betrieb vieler Container auf einem Cluster aus mehreren Servern mit automatischer Skalierung.
Controller	Oberste Schicht im Backend. Nimmt die Requests entgegen, prüft das Format und gibt sie an den Service weiter.
Cookie	Kleine Datei, die der Browser bei jedem Request an dieselbe Domain mitschickt. Hier liegt darin das JWT.
CORS	Cross-Origin Resource Sharing. Erlaubt Requests an eine andere Domain, hier nicht nötig.
CSRF	Cross-Site Request Forgery. Eine fremde Seite löst Requests im Namen des Benutzers aus. SameSite=Strict verhindert das.
Datenhoheit	Volle Kontrolle über die eigenen Daten, inklusive vollständiger Löschbarkeit.
DEFERRABLE INITIALLY DEFERRED	Die Datenbank prüft einen Constraint erst beim Commit der Transaktion.
Definition of Done	Vorab festgelegte Bedingungen, wann eine Aufgabe oder ein Arbeitspaket erledigt ist.
Deny by default	Alles ist verboten, was nicht ausdrücklich erlaubt ist. Fast alle Endpunkte brauchen ein gültiges Token.
Docker	Software, mit der man Anwendungen in Container verpackt und betreibt.
Docker Compose	Beschreibt mehrere Container in einer Datei und startet sie zusammen, hier Caddy, Frontend, Backend und Datenbank.
Dockerhub	Öffentliche Registry, in der die Container-Images liegen.
Dokumentenorientierte Datenbank	Datenbank, die Daten als JSON-ähnliche Dokumente ohne festes Schema speichert, z.B. MongoDB.
DSG	Bundesgesetz über den Datenschutz der Schweiz.
DTO	Data Transfer Object. Überträgt nur die Daten zwischen Frontend und Backend, nie die Entitäten selbst.
Endpunkt	Pfad der REST-Schnittstelle mit einer HTTP-Methode, z.B. GET /api/v1/routines.
Entität	Fachliches Objekt mit eigener ID, z.B. eine Übung. Im Backend eine Java-Klasse zu einer Tabelle.
Entscheidungsmatrix	Tabelle, in der Kandidaten anhand gewichteter Kriterien bewertet werden. Das höchste Total gewinnt.

ERM	Entity-Relationship-Modell. Diagramm der Datenbanktabellen und ihrer Beziehungen.
FA	Funktionale Anforderung. Beschreibt, was das System können muss.
Feature	Sammlung fachlich zusammenhängender Anforderungen, abgekürzt FT.
Firewall	Lässt nur Verkehr über bestimmte Ports durch, beim Server nur 22, 80 und 443.
Flyway	Werkzeug, das beim Start des Backends neue SQL-Migrationen in der richtigen Reihenfolge ausführt.
Fremdschlüssel	Spalte, die auf den Primärschlüssel einer anderen Tabelle verweist.
Frontend	Teil der Applikation, der im Browser läuft und den der Benutzer bedient.
GAF	Geschäftsanwendungsfall. Ablauf aus fachlicher Sicht, unabhängig von der Software.
GANTT-Diagramm	Balkendiagramm mit den Phasen und Arbeitspaketen auf einer Zeitachse.
Git	Versionsverwaltung, mit der alle Änderungen am Code nachvollziehbar gespeichert werden.
GitHub Actions	CI/CD-Werkzeug von GitHub. Die Workflows testen den Code und führen das Deployment aus.
Health-Check	Automatische Prüfung, ob ein Dienst läuft. Beim Backend über den Actuator unter /api/actuator/health.
HERMES	Projektmanagement-Methode der Schweizer Bundesverwaltung mit festen Phasen und Meilensteinen.
HTTP	Hypertext Transfer Protocol. Protokoll zwischen Browser und Server.
HTTP-Methode	Art eines Requests: GET liest, POST legt an, PUT ersetzt und DELETE löscht.
HTTP-Statuscode	Dreistellige Zahl, die das Ergebnis einer Anfrage angibt, z.B. 200 Erfolg oder 404 nicht gefunden.
HttpOnly	Cookie-Einstellung. JavaScript kann das Cookie nicht lesen.
HTTPS	Mit TLS verschlüsselte Variante von HTTP.
Infomaniak	Schweizer Hosting-Provider, bei dem der Server dieser Arbeit läuft.
Java	Objektorientierte Programmiersprache, in der das Backend geschrieben ist.
JPA	Jakarta Persistence API. Java-Standard, um Objekte in einer relationalen Datenbank zu speichern.
JSON	JavaScript Object Notation. Textformat, mit dem Frontend und Backend Daten austauschen.
JUnit	Testframework für Java, mit dem die Unit-Tests im Backend geschrieben sind.
JWT	JSON Web Token. Signiertes Token mit der Benutzer-ID, 8 Stunden gültig.
Kanban	Methode, um Aufgaben auf einem Board mit Spalten wie «ToDo», «In Bearbeitung» und «Erledigt» zu steuern.
Kann-Ziel	Optionales Ziel, das erst nach allen Muss-Zielen angegangen wird.

KI	Künstliche Intelligenz. Verwendet wurde vor allem Claude Opus 5.5 von Anthropic.
Leitplanke	Verbindliche Vorgabe für die Architektur, abgeleitet aus den Entscheidungen.
Let's Encrypt	Kostenlose Zertifizierungsstelle für HTTPS-Zertifikate.
Live-Protokollierung	Erfassen der Sätze direkt während des Trainings, jeder Satz wird sofort gespeichert.
Logging	Aufzeichnen von Ereignissen in Logfiles, ohne sensible Daten. Die Logs werden nach 14 Tagen gelöscht.
Login-Sperre	Nach 5 Fehlversuchen in Folge ist die Anmeldung für 15 Minuten gesperrt.
Löschkonzept	Festlegung, welche Daten wann und wie gelöscht werden.
LTS	Long Term Support. Version mit langer Update-Garantie, z.B. Java 25.
Main-Branch	Hauptzweig im Repository mit dem Stand der Produktion. Änderungen nur per Pull Request.
Material UI	Komponentenbibliothek für React, abgekürzt MUI, mit fertigen Buttons, Dialogen und Feldern.
Maven	Build-Werkzeug für Java, das die Abhängigkeiten verwaltet und die Anwendung baut.
Microservices	Architekturstil, bei dem jedes fachliche Modul ein eigener Dienst mit eigener Datenhaltung ist.
Migration	Versionierte Änderung am Datenbankschema als SQL-Datei.
Mobile-First	Die Oberfläche wird zuerst für das Smartphone gestaltet und dann für grössere Bildschirme erweitert.
Mockito	Java-Bibliothek, die in Tests Abhängigkeiten durch Attrappen ersetzt.
Modularer Monolith	Eine deploybare Einheit, intern in fachliche Module mit klaren Schnittstellen aufgeteilt.
Monorepo	Ein Repository für Backend, Frontend und Deployment.
Muss-Ziel	Ziel, das für eine erfolgreiche Diplomarbeit zwingend erfüllt sein muss.
NFA	Nicht-funktionale Anforderung, z.B. zu Sicherheit, Antwortzeit oder Bedienbarkeit.
nginx	Webserver, der im Frontend-Container die Dateien der SPA ausliefert.
Nicht-Ziel	Punkt, der bewusst nicht Teil der Diplomarbeit ist.
Node.js	Laufzeitumgebung für JavaScript, hier nur für Entwicklung und Build des Frontends.
OpenAPI	Standard, um eine REST-Schnittstelle maschinenlesbar zu beschreiben.
Opportunitätskosten	Wert der investierten Zeit, hier mit einem fiktiven Stundensatz berechnet.
Over-Engineering	Eine Lösung wird komplizierter gebaut, als es die Anforderungen verlangen.
OWASP	Organisation für die Sicherheit von Webanwendungen. Die OWASP Top 10 sind die zehn grössten Risiken.
Paywall	Bezahlschranke. Funktionen sind erst nach einem Kauf oder Abo verfügbar.

Persistenz	Dauerhaftes Speichern in der Datenbank, im Backend die unterste Schicht.
Pipeline	Automatischer Ablauf in GitHub Actions, der baut, testet und ausliefert.
PostgreSQL	Relationales Open-Source-Datenbanksystem, in dem alle Daten der Plattform liegen.
Primärschlüssel	Spalte, die jeden Datensatz einer Tabelle eindeutig kennzeichnet.
Progressives Krafttraining	Krafttraining, bei dem Gewicht oder Wiederholungen schrittweise gesteigert werden.
Proof of Concept	Früher, kleiner Nachweis, dass ein technischer Ansatz funktioniert.
Pull Request	Anfrage, Änderungen in den Main-Branch zu übernehmen. Gemergt wird nur mit grüner Pipeline.
React	JavaScript-Bibliothek für Benutzeroberflächen aus Komponenten, mit der das Frontend umgesetzt ist.
React Hook Form	Bibliothek für Formulare, die Eingaben prüft und Fehler direkt beim Feld anzeigt.
React Router	Bibliothek, die je nach Pfad die passende Seite anzeigt, ohne neu zu laden.
Relationale Datenbank	Datenbank mit Tabellen und festem Schema, die über Schlüssel verbunden sind.
Repository	Ablage eines Projekts in Git. Im Backend auch die Klasse für die Datenbankzugriffe einer Entität.
Responsive	Die Oberfläche passt sich der Bildschirmgrösse an, vom Smartphone bis zum Desktop.
REST	Representational State Transfer. Architekturstil für Schnittstellen über HTTP mit einem Pfad pro Ressource.
Reverse Proxy	Einziger Einstiegspunkt auf dem Server. Nimmt alle Anfragen an und leitet sie an den richtigen Container weiter.
Risikomatrix	Bewertung der Risiken nach Eintrittswahrscheinlichkeit und Schadenausmass.
Routine	Trainingsprogramm mit mehreren Übungen in fester Reihenfolge.
SAF	Systemanwendungsfall. Beschreibt, wie ein Akteur das System für eine Aufgabe benutzt.
SameSite=Strict	Cookie-Einstellung. Das Cookie wird nur bei Requests von der eigenen Seite mitgeschickt.
Satz	Ein Durchgang einer Übung mit Gewicht, Wiederholungen und optionaler Pausenzeit.
Schichtenmonolith	Eine deploybare Einheit, aufgeteilt in die Schichten Controller, Service und Persistenz.
Schnittstellenvertrag	Verbindliche Festlegung der Endpunkte, DTOs und Statuscodes zwischen Frontend und Backend.
Scope Creep	Der Projektumfang wächst schleichend, weil immer mehr Funktionen dazukommen.
Secret	Geheime Zugangsdaten wie DB-Passwort oder JWT-Schlüssel. Sie gehören nie ins Repository.
Secure	Cookie-Einstellung. Das Cookie wird nur über HTTPS übertragen.

Service	Mittlere Schicht im Backend mit der Geschäftslogik und den fachlichen Regeln.
Smoke-Test	Kurzer Test nach dem Deployment, ob die wichtigsten Funktionen laufen.
Soft Delete	Ein Datensatz wird nur als gelöscht markiert statt endgültig entfernt.
SPA	Single-Page-Application. Wird einmal geladen und holt die Daten danach über die REST-Schnittstelle.
Spontanes Training	Training, das ohne Planung direkt mit einer Routine gestartet wird.
Spring Boot	Java-Framework, mit dem das Backend umgesetzt ist.
Spring Security	Teil von Spring für Authentifizierung und Autorisierung, hier inklusive Passwort-Hashing und JWT.
SQL	Structured Query Language. Sprache für Abfragen und Änderungen in relationalen Datenbanken.
SSH	Secure Shell. Verschlüsselte Verbindung zum Server, auch für das Deployment.
Stakeholder	Person oder Gruppe, die am Projekt beteiligt oder davon betroffen ist.
stateless	Der Server merkt sich keine Sitzung, jeder Request wird nur über das Token geprüft.
Swagger UI	Weboberfläche zum Anzeigen und Testen der REST-Schnittstelle, nur lokal aktiv.
Systemtest	Manueller Test des ganzen Systems aus Sicht des Benutzers anhand von Testfällen.
TDD	Test Driven Development. Zuerst der Test (Red), dann der Code (Green), danach aufräumen (Refactor).
TLS	Transport Layer Security. Verschlüsselung bei HTTPS, wird beim Reverse Proxy terminiert.
Token	Digitaler Nachweis, mit dem sich ein Benutzer nach dem Login ausweist, hier ein JWT.
Trainingseinheit	Ein einzelnes Training mit einer Routine im Status «geplant», «aktiv» oder «abgeschlossen».
Transaktion	Mehrere Datenbankänderungen, die entweder alle oder gar nicht gespeichert werden.
TypeScript	JavaScript mit statischen Typen, im «strict»-Modus verwendet.
Übung	Einzelne Kraftübung mit Name, Muskelgruppe und optionaler Beschreibung.
Übungskatalog	Persönliche Sammlung aller Übungen eines Sportlers.
UI/UX	User Interface und User Experience. Gestaltung und Bedienbarkeit der Oberfläche.
Umgebungsvariable	Von aussen übergebener Wert, z.B. ein Passwort, der nicht im Code steht.
Unit-Test	Automatischer Test, der einen kleinen Teil des Codes für sich allein prüft.
Validierung	Prüfung der Eingaben. Im Frontend zur Führung, verbindlich im Backend.
Verlauf	Übersicht der abgeschlossenen Trainingseinheiten je Routine, auch Trainingshistorie genannt.

Vertical Slice	Kleiner Anwendungsfall, der einmal durch alle Schichten bis zur Datenbank umgesetzt wird.
Vite	Build-Werkzeug und Entwicklungsserver für das Frontend.
Vitest	Testframework für das Frontend.
Vorgeschlagene Sätze	Sätze aus dem letzten Training derselben Routine, mit denen die Eingabe vorbelegt wird.
VPS	Virtual Private Server. Virtueller Server bei einem Hosting-Provider, hier ein VPS Lite von Infomaniak.
WIP-Limit	Work in Progress Limit. Maximale Anzahl Aufgaben, die gleichzeitig in Bearbeitung sind.

11.3 Quellenverzeichnis

[1] Schweizerische Eidgenossenschaft, «HERMES-Projektmanagement - Big Picture,» 14 07 2026. [Online]. Available: https://www.hermes.admin.ch/de/projektmanagement/methodenueberblick.html.

[2] J. Martins, «Kanban-Methode: Definition, Prinzipien & Board erklärt,» 9 4 2026. [Online]. Available: https://asana.com/de/resources/what-is-kanban.

[3] Schweizerische Eidgenossenschaft, «Risiken managen,» 27 07 2026. [Online]. Available: https://www.hermes.admin.ch/de/projektmanagement/aufgaben/risiken-managen.html.

[4] Atlassian, «Projektrisikomanagement,» 27 07 2026. [Online]. Available: https://www.atlassian.com/de/work-management/project-management/project-risk-management.

[5] Google Gemini, «Wissensabfrage für den Buy-Case. Verwendeter Prompt: Welche sind die 3 beliebtesten Apps für Krafttraining und was blockiert die Paywall?,» 24 6 2026. [Online]. Available: gemini.google.com.

[6] A. Xu, «Monolith vs Microservices vs Modular Monoliths: What's the Right Choice,» ByteByteGo Newsletter, 20 3 2025. [Online]. Available: https://blog.bytebytego.com/p/monolith-vs-microservices-vs-modular.

[7] Spring Boot, «Spring Boot 4.1.0 available now,» Spring Boot, 10 6 2026. [Online]. Available: https://spring.io/blog/2026/06/10/spring-boot-4.

[8] D. Vega, «First-Class API Versioning in Spring Boot 4,» 25 9 2025. [Online]. Available: https://www.danvega.dev/blog/spring-boot-4-api-versioning.

[9] React, «React Versions,» React, 19 7 2026. [Online]. Available: https://react.dev/versions.

[10] mehdi, «Why Angular, Vue, and ESLint Can't Upgrade to TypeScript 7.0,» dev.to, 5 8 2026. [Online]. Available: https://dev.to/the-modern-web/why-angular-vue-and-eslint-cant-upgrade-to-typescript-70-yet-and-why-ts-71-changes-441g.

[11] OWASP, «OWASP Top 10:2025,» OWASP, 2025. [Online]. Available: https://top10.owasp.org/2025/.

[12] OWASP, «Password Storage Cheat Sheet,» OWASP, 2026. [Online]. Available: https://cheatsheetseries.owasp.org/cheatsheets/Password_Storage_Cheat_Sheet.html.

[13] Schweizerische Eidgenossenschaft, «Bundesgesetz über den Datenschutz,» 1 9 2023. [Online]. Available: https://www.fedlex.admin.ch/eli/oc/2022/491/de.

[14] Claude, «Abgrenzung Gesundheitsdaten. Verwendeter Prompt: Welche Personenbezogenen Daten würden nach DSG als Gesundheitsdaten gelten?,» Anthropic, 19 07 2026. [Online].

[15] Schweizerische Eidgenossenschaft, «Bundesgesetz über den Datenschutz, Artikel 5,» 01 09 2023. [Online]. Available: https://www.fedlex.admin.ch/eli/cc/2022/491/de#art_5.

[16] Claude, «Wissenslücke PostgreSQL Constraints. Verwendeter Prompt: Postgresql Order of cascading deletes by foreign key,» Anthropic, 15 08 2026. [Online].

[17] N. N. Thai, «What is a Spring Bean?,» Baeldung, 26 03 2026. [Online]. Available: https://www.baeldung.com/spring-bean.

[18] A. Taouam, «What actually Happens When You Start a Spring Boot Application,» Medium, 22 1 2026. [Online]. Available: https://medium.com/@ayoubtaouam/what-actually-happens-when-you-start-a-spring-boot-application-9b85421709b5.

[19] H. K. Ikkink, «IntelliJ HTTP Client: Parsing JSON Web Tokens,» jdriven, 7 03 2024. [Online]. Available: https://jdriven.com/blog/2024/03/IntelliJ-HTTP-Client-Parsing-JSON-Web-Tokens.

[20] Baeldung, «Database Migrations with Flyway,» 2 1 2026. [Online]. Available: https://www.baeldung.com/database-migrations-with-flyway.

[21] J. C. V. Sánchez, «Spring Boot Actuator,» Baeldung, 24 08 2025. [Online]. Available: https://www.baeldung.com/spring-boot-actuators.

[22] Infomaniak, «VPS Lite Linux und Windows,» Infomaniak, [Online]. Available: https://www.infomaniak.com/de/hosting/vps-lite.

[23] Caddy, «Reverse proxy quick-start,» Caddy, [Online]. Available: https://caddyserver.com/docs/quick-starts/reverse-proxy.

11.4 Tabellenverzeichnis

Tabelle 1: Risikobewertung 14
Tabelle 2: Bewertung der Stakeholder 18
Tabelle 3: Vergleich bestehender Apps 19
Tabelle 4: Projektphasen 21
Tabelle 5: Arbeitspakete Initialisierung 23
Tabelle 6: Arbeitspakete Planung 24
Tabelle 7: Arbeitspakete Vorbereitung 24
Tabelle 8: Arbeitspakete Anforderungsanalyse 25
Tabelle 9: Arbeitspakete Konzeption 25
Tabelle 10: Arbeitspakete Entwicklung 26
Tabelle 11: Arbeitspakete Deployment 27
Tabelle 12: Arbeitspakete Abschluss 28
Tabelle 13: Aufwand pro Phase 28
Tabelle 14: Vorkenntnisse Backend und Datenbanken 29
Tabelle 15: Vorkenntnisse Frontend 30
Tabelle 16: Vorkenntnisse Infrastruktur und Tools 30
Tabelle 17: Architekturtreiber 31
Tabelle 18: Entscheidungsmatrix Persistenzansatz 33

Tabelle 19: Entscheidungsmatrix Backend-Stil 34
Tabelle 20: Entscheidungsmatrix Frontend-Ansatz 36
Tabelle 21: Entscheidungsmatrix Deployment-Topologie 37
Tabelle 22: Leitplanken 40
Tabelle 23: Entscheidungsmatrix Backend-Framework 41
Tabelle 24: Entscheidungsmatrix Frontend-Framework 43
Tabelle 25: Entscheidungsmatrix Hosting Provider 44
Tabelle 26: Ergänzende Technologien Backend 45
Tabelle 27: Ergänzende Technologien Frontend 45
Tabelle 28: Ergänzende Technologien Infrastruktur 46
Tabelle 29: Schutzmassnahmen nach OWASP 47
Tabelle 30: Schutzobjekte und Schutzbedarf 47
Tabelle 31: Personendaten 49
Tabelle 32: Teststufen 51
Tabelle 33: Werkzeuge CI/CD 52
Tabelle 34: Muss-Ziele 54
Tabelle 35: GAF1 Benutzerkonto führen 55
Tabelle 36: GAF2 Benutzerkonto auflösen 56
Tabelle 37: GAF3 Trainingsprogramm zusammenstellen 56
Tabelle 38: GAF4 Training planen 57
Tabelle 39: GAF5 Training durchführen und protokollieren 57
Tabelle 40: GAF6 Trainingsfortschritt nachvollziehen 57
Tabelle 41: Ausgeschlossene GAF 58
Tabelle 42: SAF1 Registrieren 60
Tabelle 43: SAF2 Anmelden 61
Tabelle 44: SAF3 Profil verwalten 62
Tabelle 45: SAF4 Konto löschen 63
Tabelle 46: SAF5 Übung verwalten 63
Tabelle 47: SAF6 Trainingsroutine verwalten 64
Tabelle 48: SAF7 Training einplanen 65
Tabelle 49: SAF8 Training durchführen und protokollieren 66
Tabelle 50: SAF9 Trainingshistorie einsehen 67
Tabelle 51: Anforderungen pro Feature 68
Tabelle 52: FT1 Benutzerverwaltung & Authentifizierung 69
Tabelle 53: FA zu FT1 70
Tabelle 54: NFA zu FT1 71
Tabelle 55: FT2 Übungskatalog 72
Tabelle 56: FA zu FT2 72
Tabelle 57: FT3 Trainingsroutinen 73
Tabelle 58: FA zu FT3 73
Tabelle 59: FT4 Trainingsplanung 74
Tabelle 60: FA zu FT4 74
Tabelle 61: NFA zu FT4 75
Tabelle 62: FT5 Live-Protokollierung 75
Tabelle 63: FA zu FT5 75
Tabelle 64: NFA zu FT5 76
Tabelle 65: FT6 Trainingshistorie 77
Tabelle 66: FA zu FT6 77

Tabelle 67: NFA zu FT6 78
Tabelle 68: FT7 Responsive Web-App & Querschnitt 78
Tabelle 69: NFA zu FT7 78
Tabelle 70: Löschverhalten der Fremdschlüssel 84
Tabelle 71: Umsetzung der Löschung 85
Tabelle 72: Eindeutigkeit 86
Tabelle 73: Prüfregeln 86
Tabelle 74: Indizes 87
Tabelle 75: Stammdaten Muskelgruppen 87
Tabelle 76: Stammdaten Status 87
Tabelle 77: Regeln im Service 91
Tabelle 78: Statusmodell der Trainingseinheit 92
Tabelle 79: Statuscodes 94
Tabelle 80: Endpunkte 94
Tabelle 81: Request-DTOs 97
Tabelle 82: Response-DTOs 99
Tabelle 83: Fehlerformat 100
Tabelle 84: Screens 105
Tabelle 85: Wichtige Elemente 109
Tabelle 86: Unit-Tests Backend 112
Tabelle 87: Manuelle Systemtests Backend 114
Tabelle 88: Umgebungsvariablen 115
Tabelle 89: Öffentliche Endpunkte 118
Tabelle 90: Geschützte Endpunkte 118
Tabelle 91: Manuelle Systemtests Frontend 121
Tabelle 92: Manuelle Tests Status Trainingseinheit 123
Tabelle 93: Testfälle Löschkonzept 125
Tabelle 94: Unit-Tests Frontend 126
Tabelle 95: Weitere Befehle 128
Tabelle 96: Ordnerstruktur 128
Tabelle 97: Smoke-Tests 136
Tabelle 98: Testergebnisse 140
Tabelle 99: Erfolgskontrolle 141
Tabelle 100: Glossar 145
Tabelle 101: Vorzeigetermin 1 157
Tabelle 102: Vorzeigetermin 2 158

11.5 Bildverzeichnis

Abbildung 1: Risikomatrix 14
Abbildung 2: GANTT Diagramm 22
Abbildung 3: Kanban-Board in Jira 23
Abbildung 4: Systemkontext 32
Abbildung 5: Bausteinsicht 38
Abbildung 6: Technologie-Stack 46
Abbildung 7: Geschäftsanwendungsfälle 55
Abbildung 8: Zerlegung der GAF in SAF 59

Abbildung 9: Systemanwendungsfälle 60
Abbildung 10: Zuordnung der SAF zu den Features 69
Abbildung 11: ERM 84
Abbildung 12: Paketdiagramm 88
Abbildung 13: Klassendiagramm Benutzerverwaltung 88
Abbildung 14: Klassendiagramm Übungskatalog 89
Abbildung 15: Klassendiagramm Trainingsroutinen 89
Abbildung 16: Klassendiagramm Trainingsprotokoll Controller und Service 90
Abbildung 17: Klassendiagramm Trainingsprotokoll DTOs 90
Abbildung 18: Klassendiagramm Trainingsprotokoll Persistenz 90
Abbildung 19: Klassendiagramm Querschnitt 91
Abbildung 20: Aktivitätsdiagramm Training einplanen 101
Abbildung 21: Aktivitätsdiagramm Training durchführen 102
Abbildung 22: Sequenzdiagramm Login 103
Abbildung 23: Sequenzdiagramm Anfrage mit Token 104
Abbildung 24: Zustandsdiagramm Trainingseinheit 105
Abbildung 25: Navigation zwischen den Screens 108
Abbildung 26: Pull Request mit erfolgreichen Tests 110
Abbildung 27: Swagger UI 120
Abbildung 28: Startbildschirm 130
Abbildung 29: Live Training 131
Abbildung 30: Neue Übung 131
Abbildung 31: Routine bearbeiten 132
Abbildung 32: Verlauf 132
Abbildung 33: Registrieren 133
Abbildung 34: Anmelden 133
Abbildung 35: Übersicht 133
Abbildung 36: Konto 133
Abbildung 37: Deploy Workflow 136

11.6 Vorzeigetermine

11.6.1 Termin 1

Tabelle 101: Vorzeigetermin 1

Anlass	1. Vorzeigetermin (Standortbestimmung) gemäss Richtlinien Diplomarbeit
Diplomarbeit	Digitale Plattform zur Trainingsverwaltung
Datum	Montag, 10. August 2026
Zeit	11:00 – 12:00 Uhr
Ort / Form	Microsoft Teams (Onlinebesprechung)
Teilnehmende	Dominic Merz (Diplomcoach), Lennard Bernet (Diplomand)
Protokollführung	Lennard Bernet
Nächster Termin	Montag, 12. Oktober 2026 (2. Vorzeigetermin)

Standortbestimmung in Bezug auf die Zielsetzung

Es wurde der aktuelle Stand der Diplomarbeit dem Diplomcoach vorgestellt. Der erste Eindruck der bisher erarbeiteten Inhalte war durchaus positiv. Der Diplomcoach meinte, dass der Stand der Arbeit absolut in Ordnung ist und ich aktuell voll auf Kurs liege. Die erarbeiteten Arbeitspakete sind gut so und können gebraucht werden für die weiteren Arbeiten an der Diplomarbeit. Der Zweck des Termins war es, den aktuellen Stand zu überprüfen und frühzeitig grosse Abweichungen zu erkennen.

Offene Frage Microsite

Es wurde die Frage eingebracht, wie die Abgabe der Microsite abläuft, da dazu noch gar keine Arbeiten erfolgt sind. Folgende Punkte wurden zu diesem Thema geklärt:

- Man geht davon aus, dass der Link zur Microsite erst am 11. September 2026 verschickt wird und die Bearbeitung erst ab diesem Zeitpunkt möglich sein wird.
- Mit diesem Link gelangt man anschliessend auf die Microsite, welche man selbstständig erarbeiten kann. Abschliessend gibt es dann dort einen weiteren Link, über den man die Microsite dem Diplomcoach zur Prüfung freigeben kann.
- Sobald der Diplomcoach zufrieden ist, wird die Microsite freigegeben
- Die Microsite und der Kompetenzdialog müssen nicht in die Diplomarbeit und somit in den Zeitplan aufgenommen werden, sondern sind separate Prozesse.

Hinweise zu Aufbau und Struktur der Dokumentation

Folgende Hinweise gab der Diplomcoach zur weiteren Ausarbeitung der Dokumentation

- Anhand des Bewertungsrasters die Struktur und Inhalte immer wieder kontrollieren. Somit stellt man sicher, dass alle relevanten Punkte berücksichtigt werden.
- Eine Entscheidungsmatrix einfliessen lassen. Ein gutes Beispiel ist die Wahl der Architektur.
- Getroffene Entscheidungen auch begründen. Die verschiedenen Möglichkeiten einander gegenüberstellen.
- Bei den Arbeitspaketen Reserven einplanen. Die Arbeitspakete mit einem Gantt-Diagramm visualisieren.

Abgabe, Präsentation und Qualifikationsprofil

- Nicht vergessen am Qualifikationsprofil zu arbeiten. Dort immer wieder mal weitermachen.
- Die Abgabe erfolgt als PDF am Abgabetermin. Anschliessend erwartet der Diplomcoach die Diplomarbeit zusätzlich als gedruckte Version.
- Die Präsentation wird in Mundart durchgeführt. Ebenfalls die anschliessende Fragerunde ist in Mundart.

Fazit und weiteres Vorgehen

Als Fazit hielt der Diplomcoach fest, dass der Stand der Arbeit vollständig in Ordnung ist. Zusätzlich wurden noch einige Tipps zur weiteren Bearbeitung abgegeben. Nach Einschätzung des Diplomcoachs liegt die Arbeit aktuell auf Kurs. Der bereits vereinbarte nächste Termin vom 5. Oktober 2026 wurde auf den 12. Oktober verschoben. Bis dahin können offene Fragen jederzeit per Chat mit dem Diplomcoach geklärt werden.

11.6.2 Termin 2

Tabelle 102: Vorzeigetermin 2

Anlass	2. Vorzeigetermin (Standortbestimmung) gemäss Richtlinien Diplomarbeit
Diplomarbeit	Digitale Plattform zur Trainingsverwaltung
Datum	Montag, 12. Oktober 2026
Zeit	11:00 – 12:00 Uhr
Ort / Form	Microsoft Teams (Onlinebesprechung)
Teilnehmende	Dominic Merz (Diplomcoach), Lennard Bernet (Diplomand)
Protokollführung	Lennard Bernet
Nächster Termin	Mittwoch, 28. Oktober 2026 (Abgabe)

Allgemeine Standortbestimmung

Der ursprüngliche Termin vom 5. Oktober wurde auf den 12. Oktober verschoben. Durch eine Verletzung im rechten Unterarm und persönliche Engpässe ist die Diplomarbeit im Verzug im Zeitplan. Dies wurde mit dem Diplomcoach besprochen, um den Fokus auf die nötigsten Themen einzugrenzen. Das Ziel während der Besprechung war es, so gut wie möglich festzulegen, was inhaltlich noch fehlt und wie dies zu dokumentieren ist.

Der aktuelle Stand ist stark und es wurden viele wichtige Punkte schon abgedeckt. Der Aufbau von den Technologieentscheidungen, Cybersecurity und Datenschutzkonzepte über zu den Anforderungen und technische Konzepte soll so beibehalten werden. Jetzt den Fokus darauflegen, den Rest der Dokumentation fertigzustellen. Erst wenn man Zeit hat gewisse Punkte komplett ausfeilen. Die vollständige Dokumentation ist wichtiger als ein perfekt bugfreier lauffähiger Prototyp. Offene Arbeitspakete sollen auf hoher Flugebene vorangebracht werden, sodass die Definitions of Done gerade so erreicht werden. Erst anschliessend jedes Detail noch komplett ergänzen, wo es wirklich nötig ist.

Im Kapitel Entwicklung ist die Dokumentation des Testings das Wichtigste.

Abgabeumfang besprochen

Der Diplomcoach hat nochmals darauf verwiesen, dass bei der Abgabe ein gebundenes Exemplar gewünscht ist. Mit dem Diplomexperten muss das noch festgelegt werden.

Der Umfang der Microsite ist in etwa ein Management Summary. Gepaart mit Bildern ist es eine Art Visitenkarte gegen aussen.